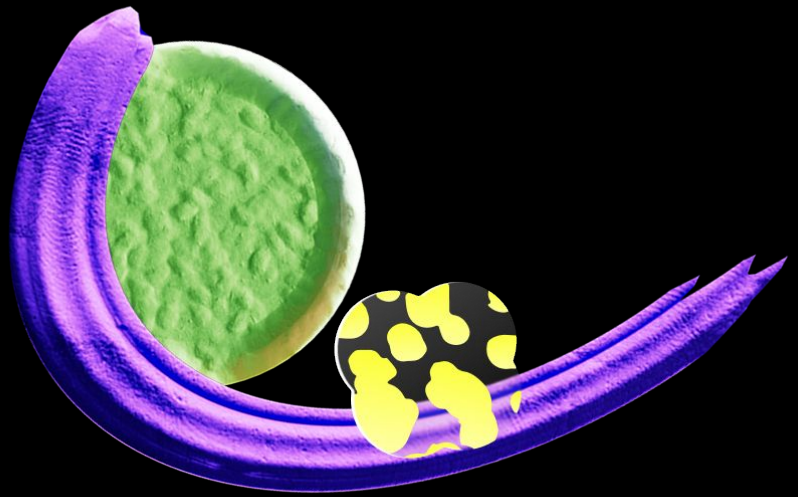
Observability микрофронтендов

Дарья Саенко
Frontend Engineer

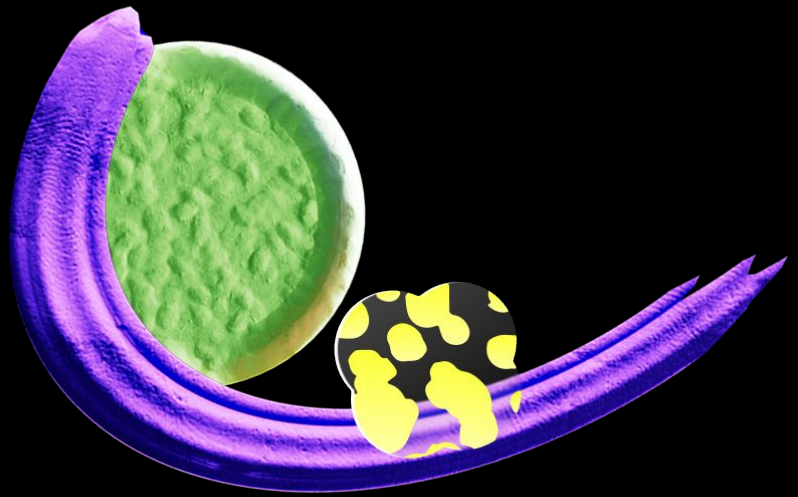
Спикер

Дарья Саенко
Frontend Engineer

Работаю в Авито, в юните Frontend Architecture, разрабатываю решения для микрофронтендов и рантайма.



Чем занимается юнит Frontend Architecture в Авито



- Проектируем архитектуру фронтенда для упрощения продуктовой разработки
- Разрабатываем инструменты DX и CI, подходы и инструменты тестирования
- Развиваем UI-kit и отвечаем за консистентность всего UI/UX продукта Авито

О чём расскажу в докладе

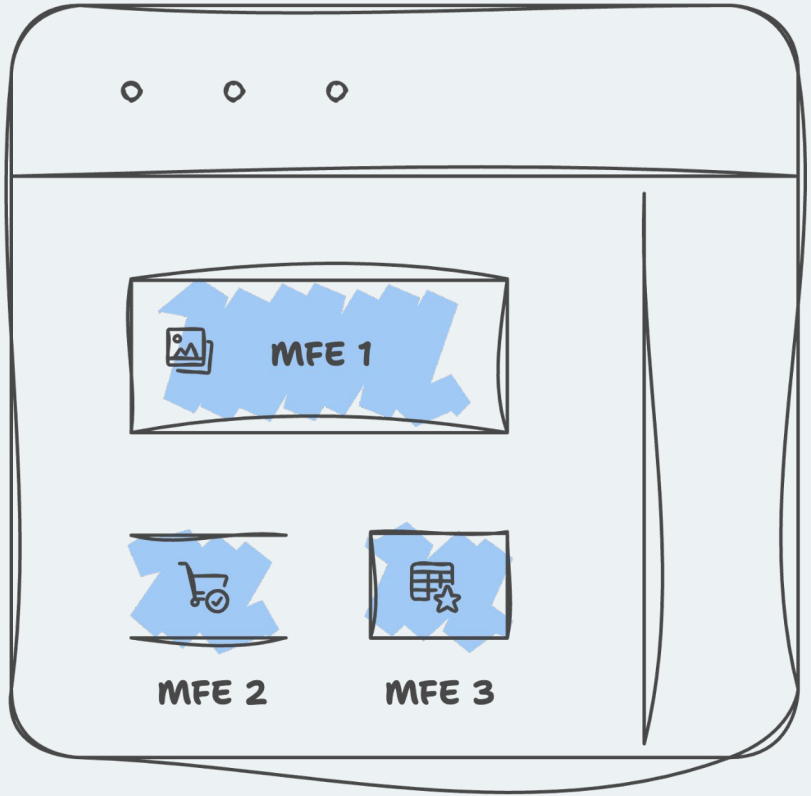
- Из чего состоит наша **система observability**
- Какие инструменты используем
- Что логируем, какие метрики пишем и почему
- Как распространяем лучшие практики на множество микрофронтендов

Что такое микрофронтенды

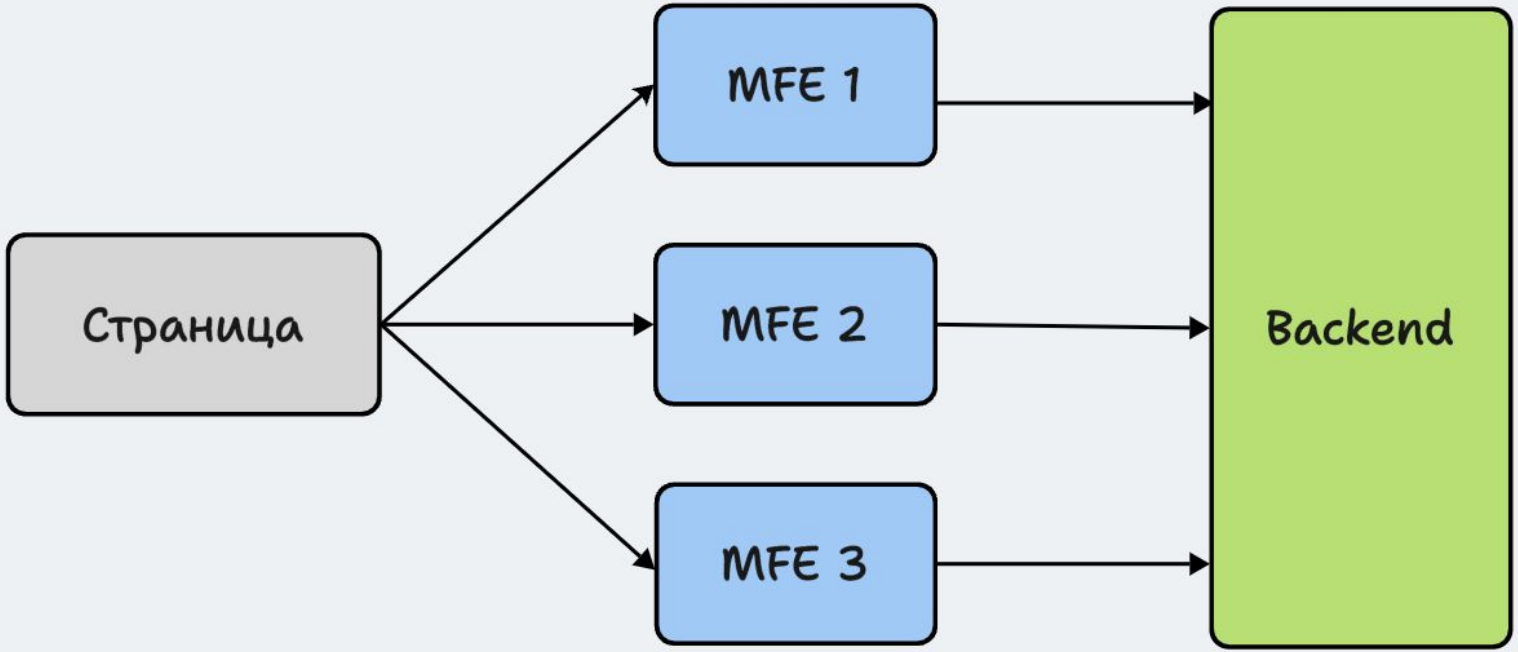
Микрофронтенды

Это архитектурный подход, при котором веб-приложение разбивается на независимые части.

Эти части обычно разрабатываются разными командами и релизятся отдельно друг от друга.



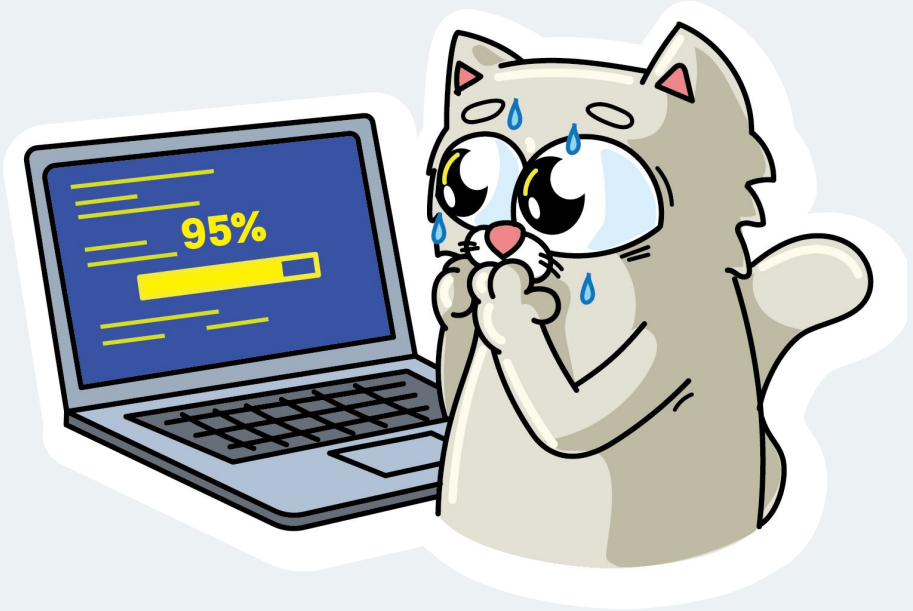
Архитектура микрофронтендов



Зачем нам микрофронтенды

До микрофронтендов у нас были единые **монолит-приложения**,
но они имели ряд минусов:

- долгий CI/CD;
- низкая отказоустойчивость;
- сложность детекции проблем;
- сложность поддержки.



Микрофронтенды

Снизили TTM

в 5 раз

сократили время деплоя

за счет деплоя меньшего объема кода.

в 10 раз

выросло число релизов

за счет релизов по требованию вместо релизов по расписанию.

~в 7 раз

сократили число E2E

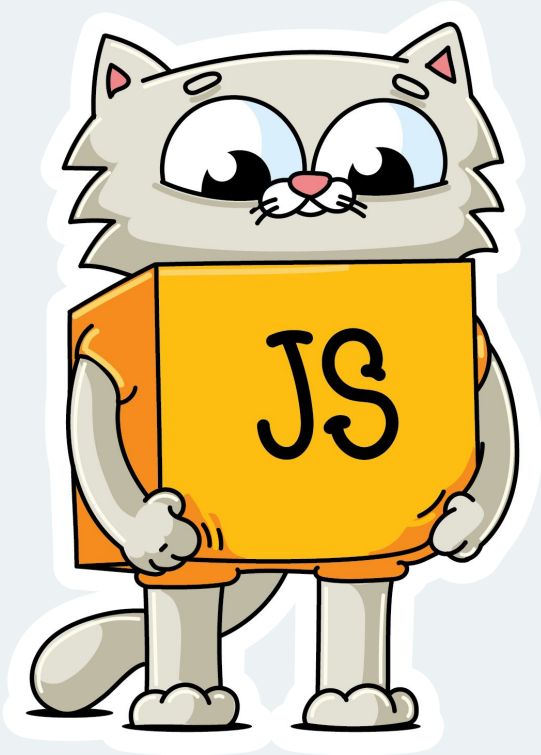
запускаемых на коммите.

Микрофронтенды позволили

Повысить надежность сайта
за счет уменьшения связанности с основным приложением и контрактной системе взаимодействия.

Повысить скорость детекции
за счет меньшего количества задач в релизах и чёткого разделения ответственности.

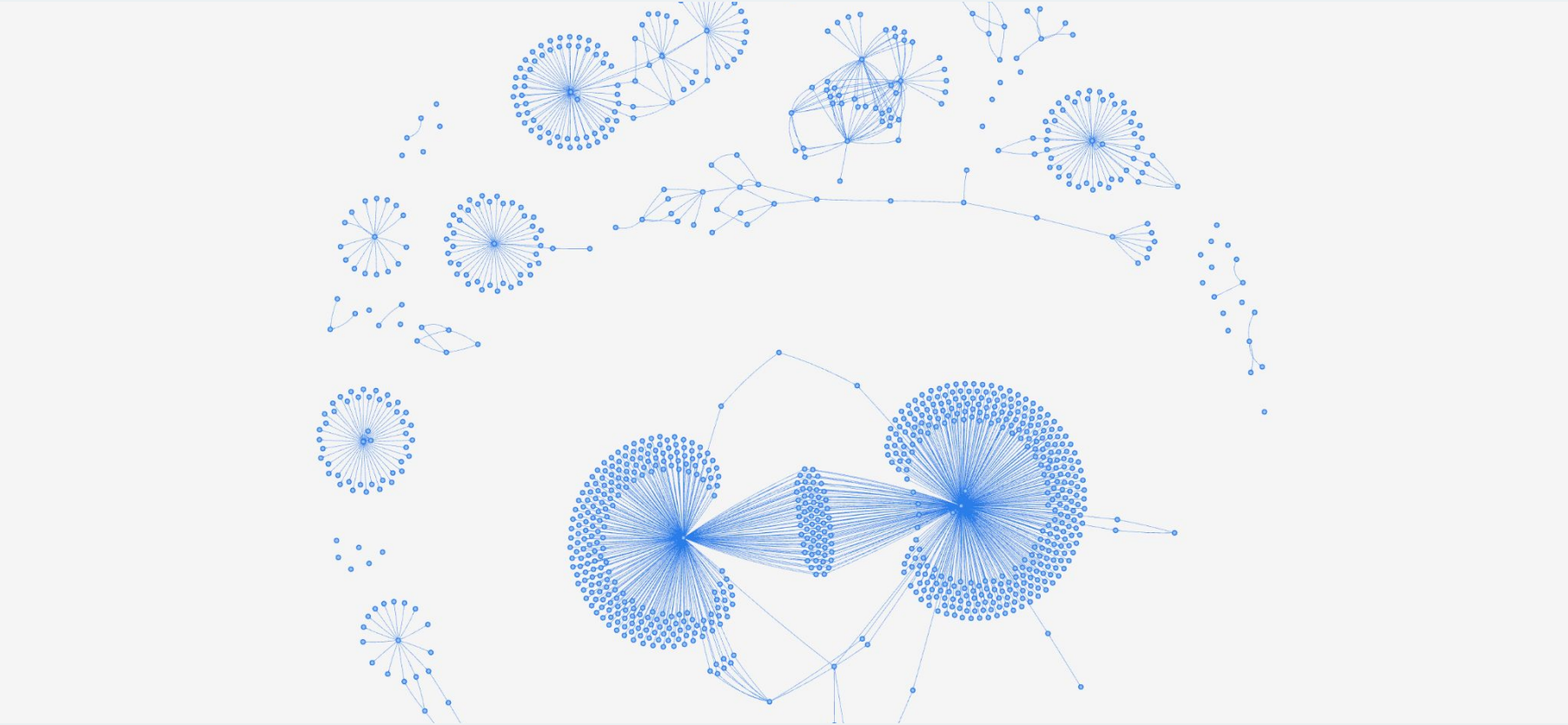
Уменьшить связанность команд
за счёт ограничения доступа к внешним ресурсам.



Микрофронтенды

Как это устроено у нас

- Микрофронтенды **работают через внутреннее решение** на базе Module Federation.
- Микрофронтенды рендерятся **на сервере и клиенте.**
- Микрофронтенд принадлежит сервису на NodeJs; **сервис может иметь один и более микрофронтендов.**
- Микрофронтенды **могут встраиваться** в разные приложения (хосты).



В Авито ~1000 микрофронтендов

Как следить за тысячей микрофронтендов?

Что такое observability

Observability

Это подход, позволяющий в реальном времени получать представление о состоянии системы, чтобы оперативно находить, анализировать и устранять проблемы.

Как достичь надежности

01

Предотвращать проблемы

02

Реагировать на проблемы

03

Измерять успех

Как достичь надежности

01

Предотвращать проблемы

Канареечные релизы

02

Реагировать на проблемы

Алерты

03

Измерять успех

NFR

Как достичь надежности

Термины

Канареечный релиз – метод развертывания сервиса, при котором новая версия сначала становится доступной небольшой группе пользователей.

Алерты – автоматические уведомления о критических событиях и изменениях показателей, требующие внимания разработчика.

NFR – нефункциональные требования, относящиеся прямо или косвенно к производительности сервиса.

Как предотвращаем инциденты

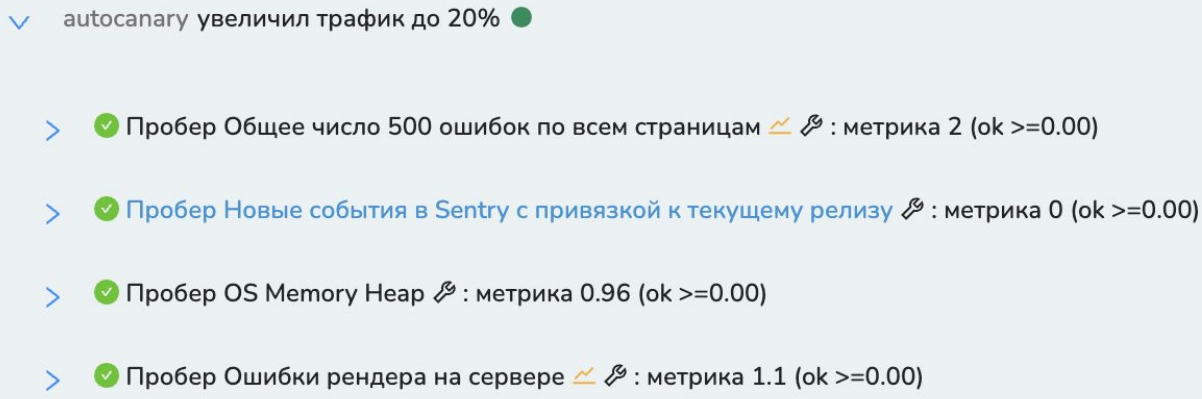
Канареечные релизы

Деплоим **критичные сервисы** через канареечные релизы с автоматическим увеличением процента и автоматическим откатом в случае проблем на основе проберов.

Как предотвращаем инциденты

Канареечные релизы

Пробер – проверка параметра, которая периодически запускается во время деплоя и проверяет установленные границы.

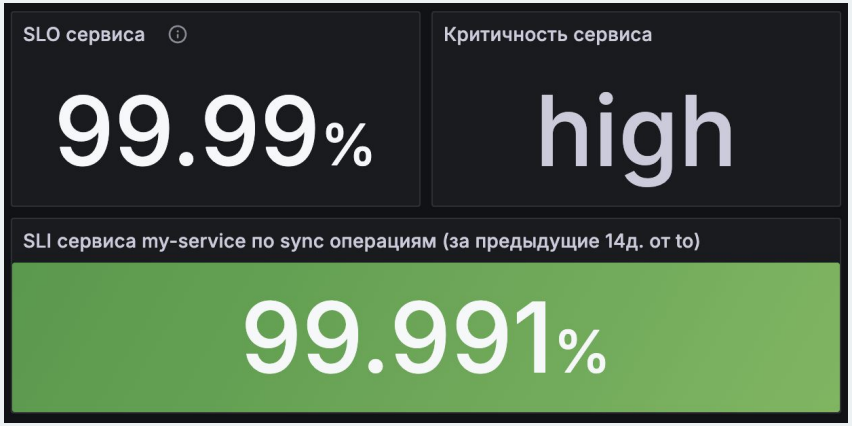


Как измеряем успех
NFR

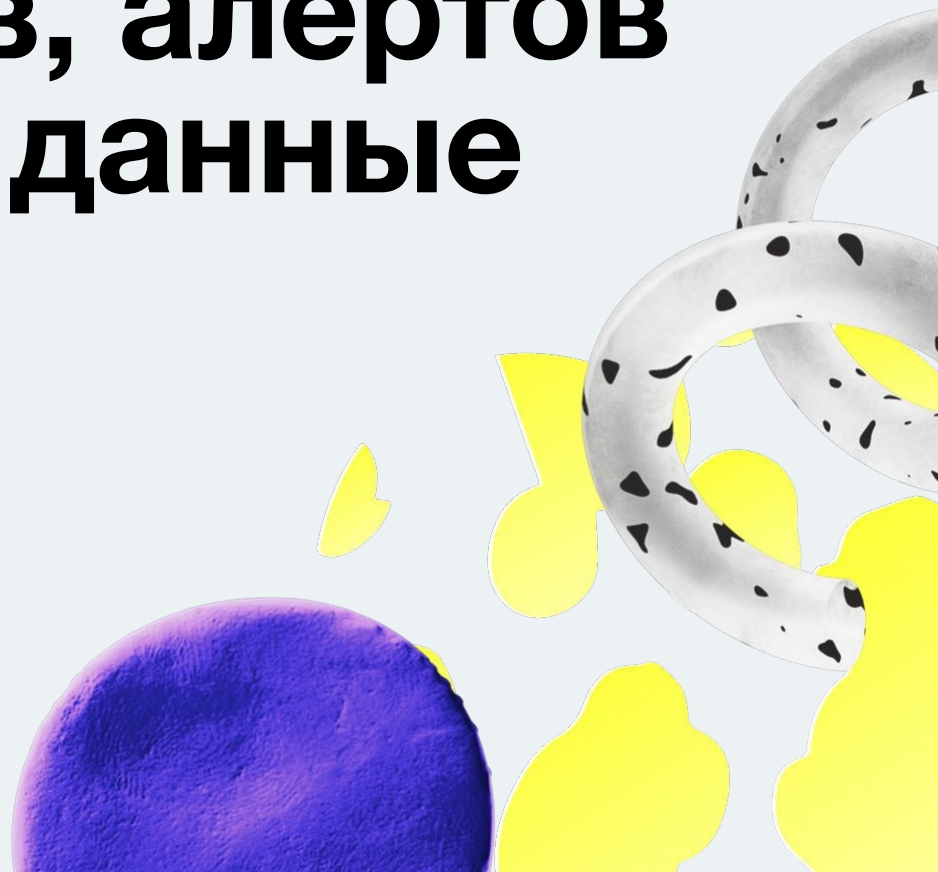
Описываем требования к производительности и надежности, а затем сравниваем с реальными показателями.

SLO – целевой уровень качества.

SLI – текущий уровень качества.



Для проберов, алертов и NFR нужны данные



Из чего состоит observability

01

Логи

Текстовые события, обычно с контекстом.

02

Метрики

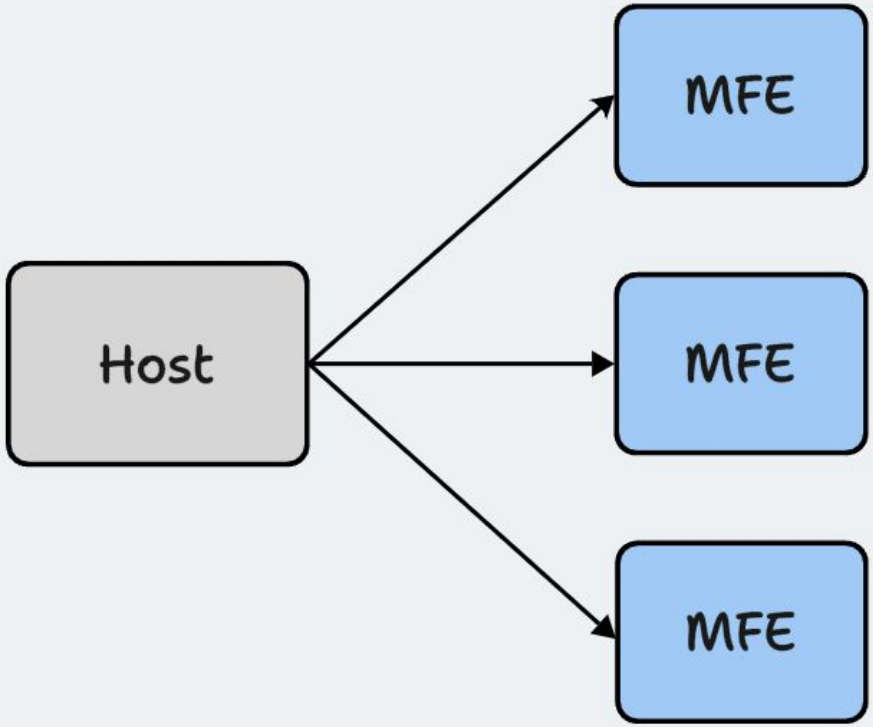
Численные значения, которые можно агрегировать и отображать.

03

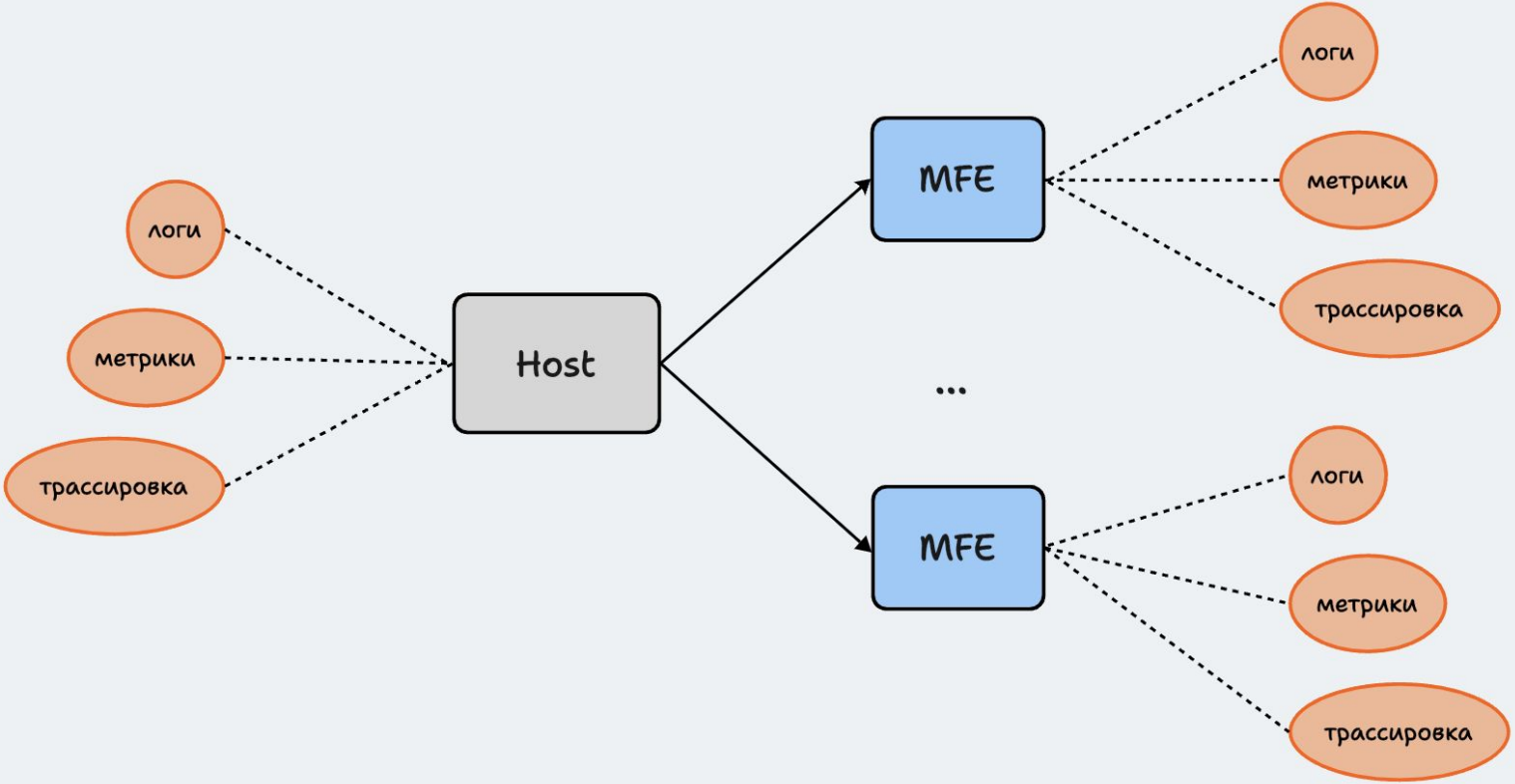
Трассировка

Хронология вызовов между компонентами системы.

Архитектура микрофронтендов

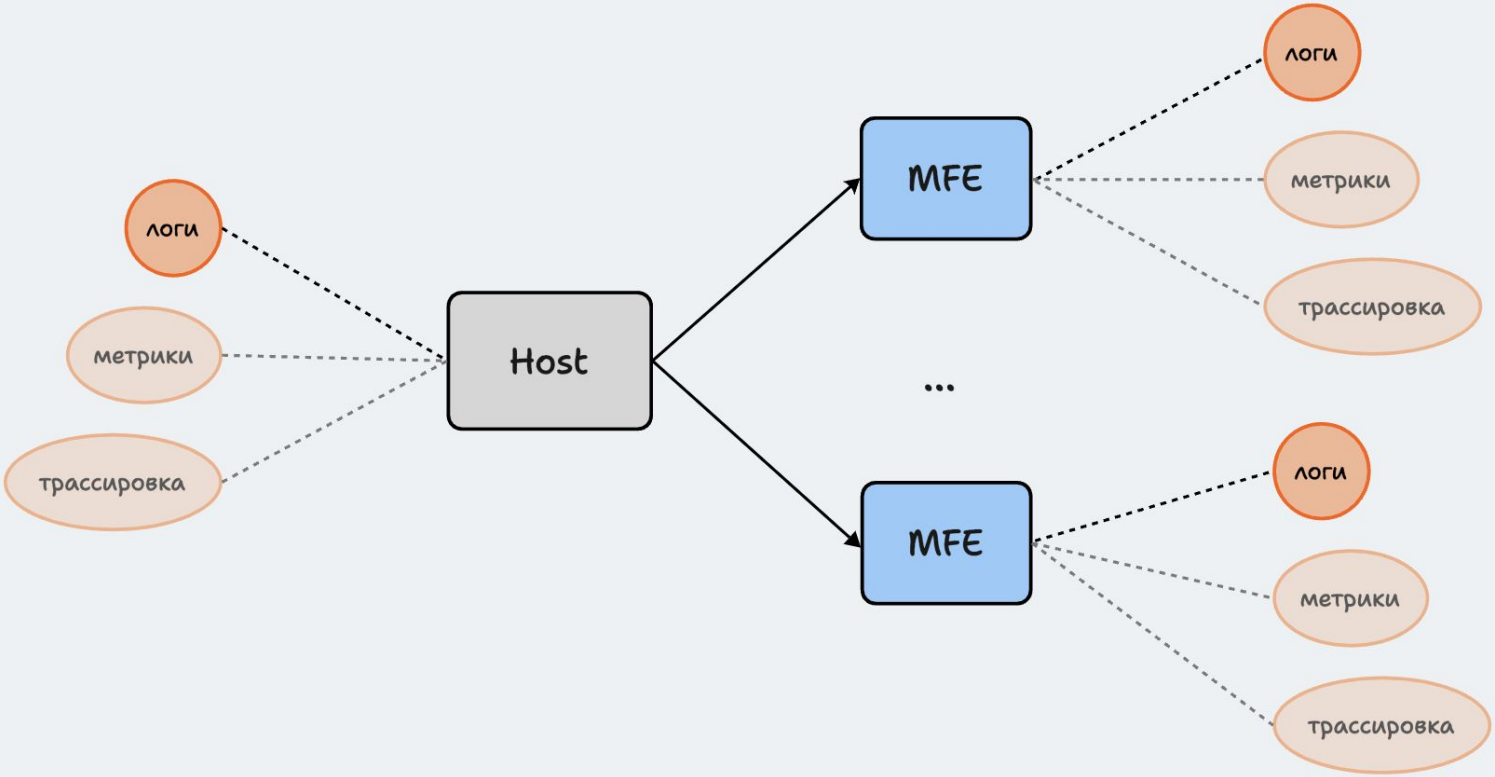


Observability на схеме



Observability. Логи

Логи

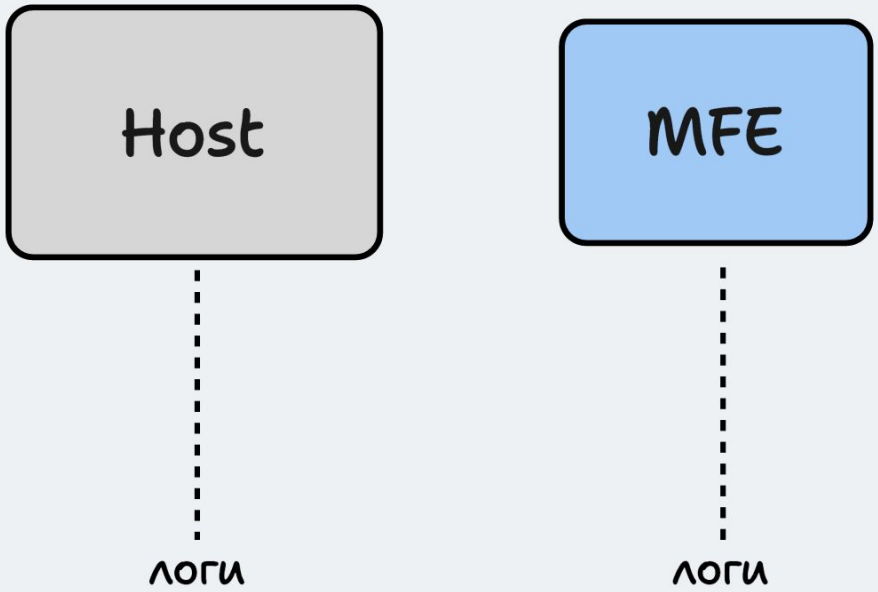

логи
метрики
трассировка
Host
MFE
...
MFE
логи
метрики
трассировка
логи
метрики
трассировка

Логи

Откуда писать

Логи микрофронтенда пишем на сервере и клиенте из сервиса, в котором находится микрофронтенд.

Хост пишет только собственные логи.



Логи

Откуда писать – альтернативный вариант

Если писать логи о микрофронтенде **из хоста**:

- система будет менее надежной;
- небезопасно;
- нет гарантии, что хост собирает логи.

Если что-то сломается в хосте, не будет информации о его микрофронтендах

Есть риск потерять логи

Логи

Что писать

01 Ошибки

Но исключаем те, которые невозможно исправить.

02 Данные для отладки

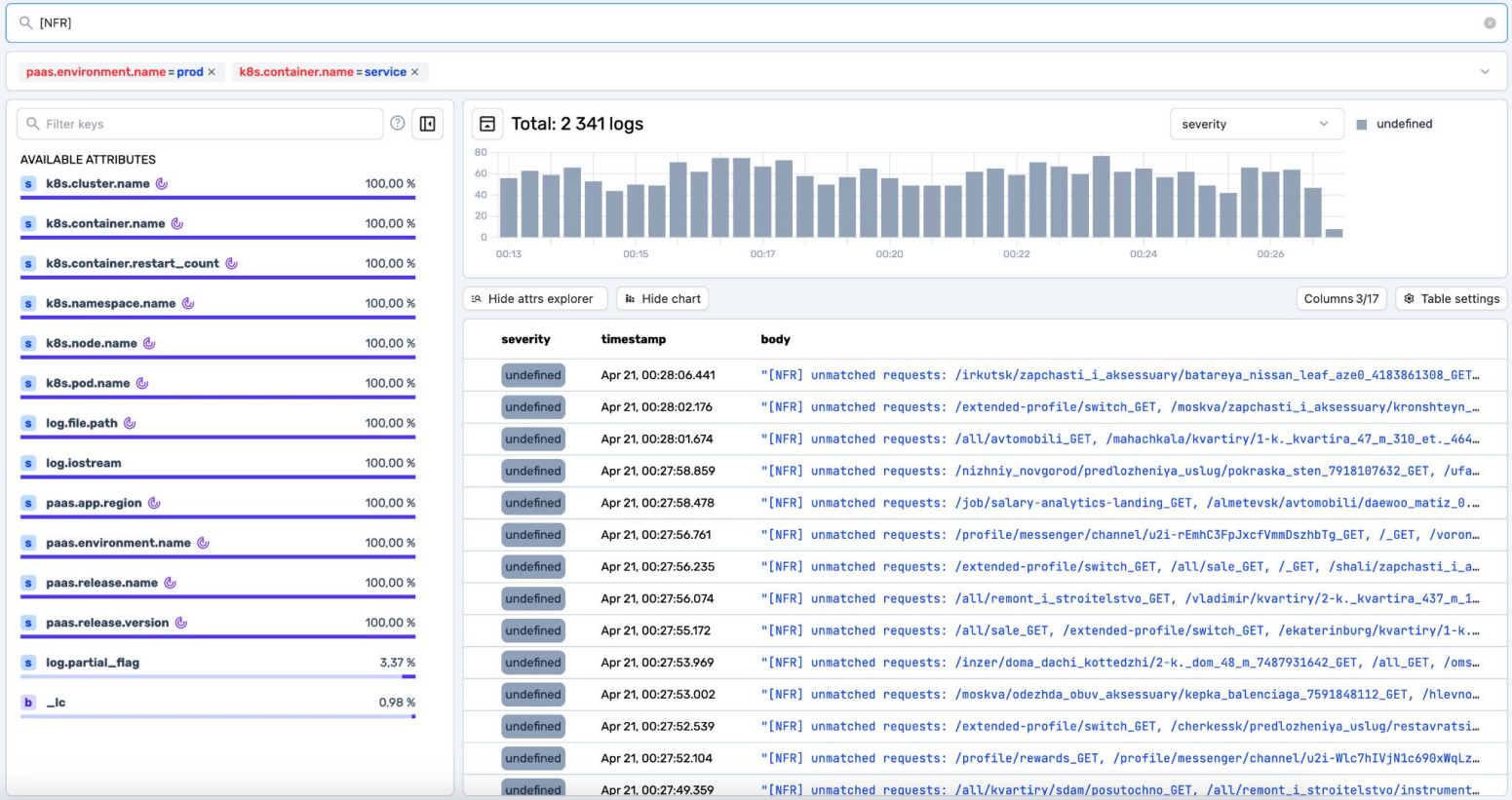
Вспомогательная информация для отладки, если требуется.

Логи

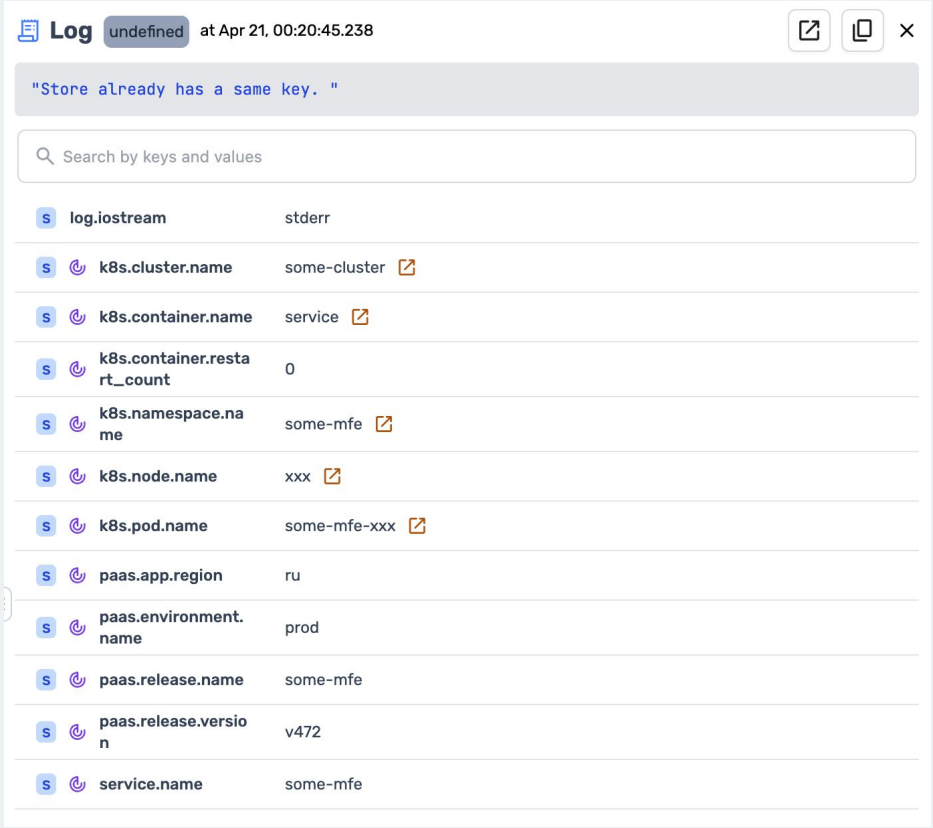
Инструменты для логов

- OpenTelemetry – современный открытый стандарт для сбора логов, метрик и трассировок.

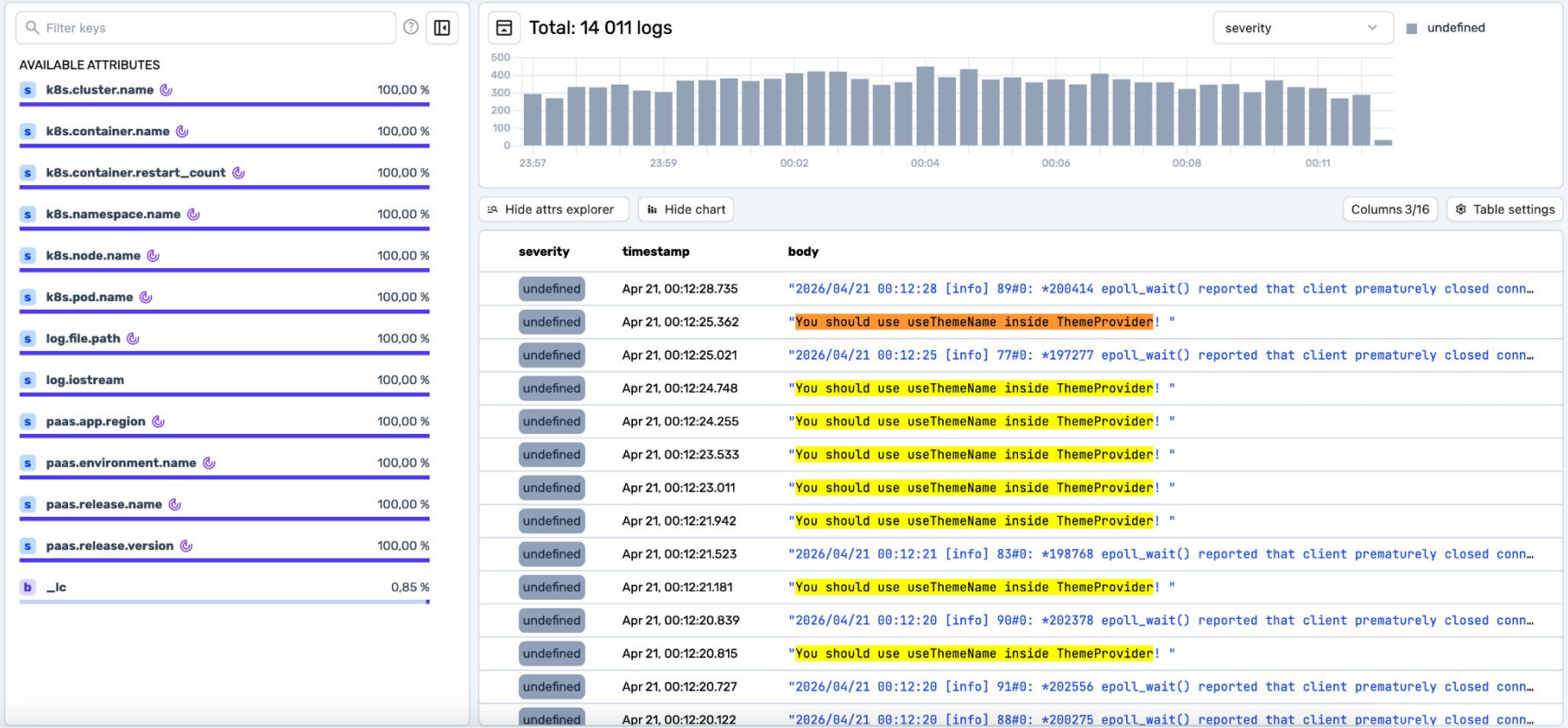
- ClickHouse — масштабируемое столбцовое хранилище.



Пример того, когда могут быть полезны



Пример того, когда логи не помогают



Пример того, когда логи мешают

```
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
Store already has a same key.
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
You should use useThemeName inside ThemeProvider!
```

Но это все равно лучше, чем смотреть логи в console

Логи

Инструменты для мониторинга ошибок

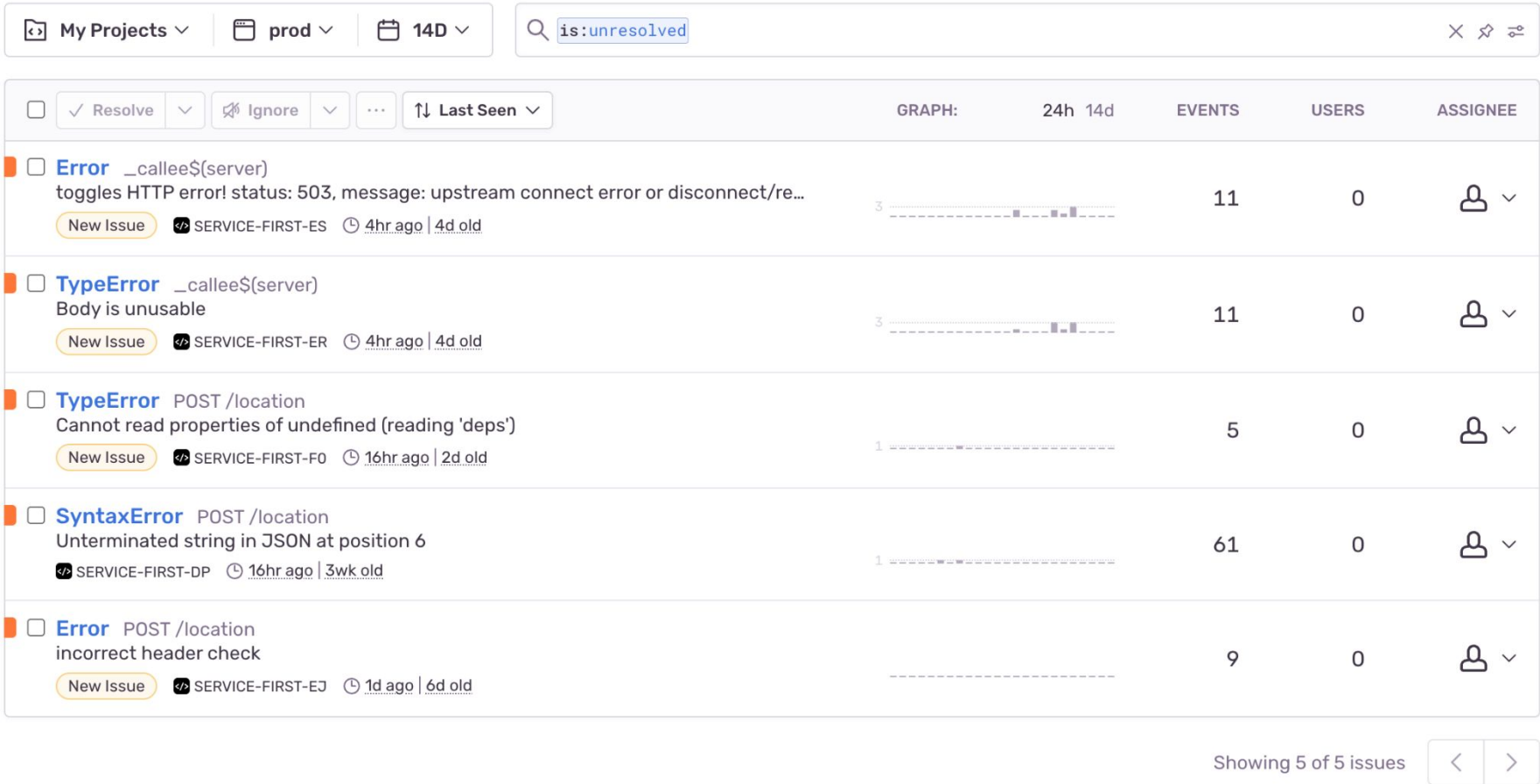
Sentry – платформа мониторинга производительности и отслеживания ошибок в режиме реального времени.

Sentry Microfrontends — методология настройки мониторинга ошибок, при которой микрофронтенды интегрируются с платформой Sentry.

Sentry Microfrontends

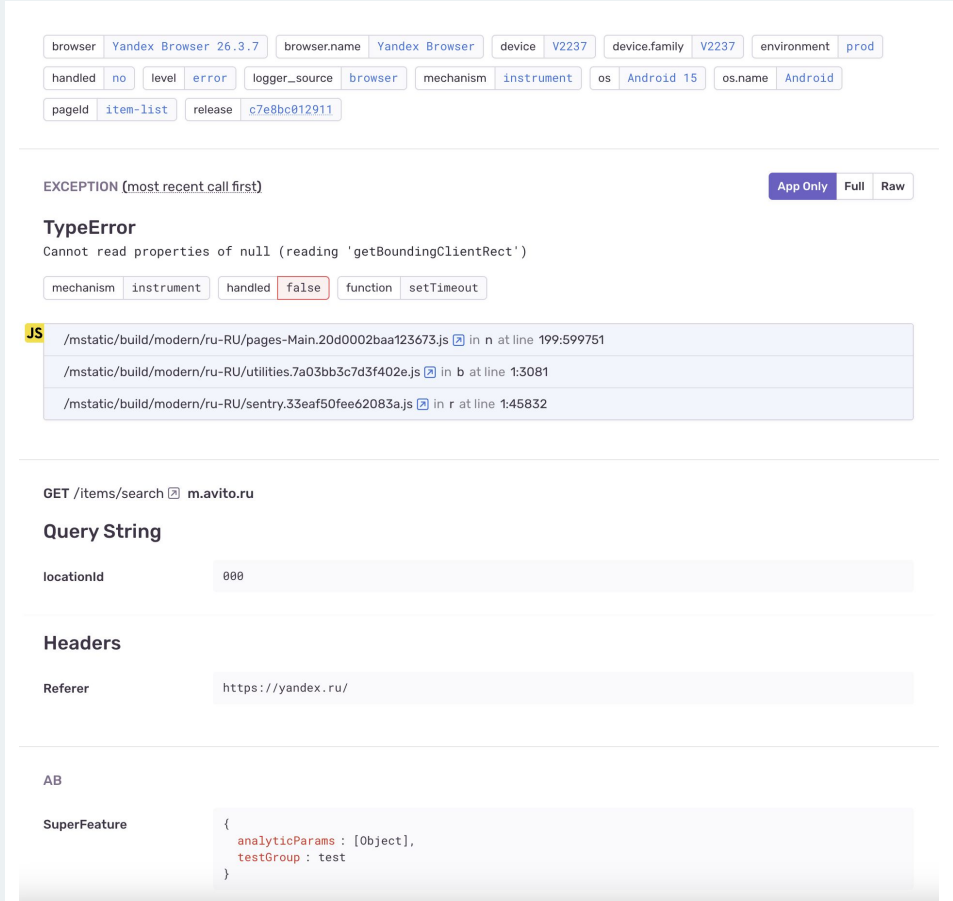
Зачем нужен инструмент:

- нет ошибок из других сервисов;
- в одном проекте Sentry собираем ошибки и с сервера, и с клиента;
- команды могут настраивать свои алерты в Sentry.


My Projects
prod
14D
is:unresolved
Resolve
Ignore
Last Seen
GRAPH:
24h 14d
EVENTS
USERS
ASSIGNEE
Error _callee$(server)
toggles HTTP error! status: 503, message: upstream connect error or disconnect/re...
New Issue
SERVICE-FIRST-ES
4hr ago | 4d old
11
0
TypeError _callee$(server)
Body is unusable
New Issue
SERVICE-FIRST-ER
4hr ago | 4d old
11
0
TypeError POST /location
Cannot read properties of undefined (reading 'deps')
New Issue
SERVICE-FIRST-F0
16hr ago | 2d old
5
0
SyntaxError POST /location
Unterminated string in JSON at position 6
SERVICE-FIRST-DP
16hr ago | 3wk old
61
0
Error POST /location
incorrect header check
New Issue
SERVICE-FIRST-EJ
1d ago | 6d old
9
0
Showing 5 of 5 issues

Мониторинг ошибок

Пишем в ошибку всю дополнительную информацию, которая поможет при отладке.



Мониторинг ошибок

Предотвращать проблемы

Проберы для канареечных релизов

- **New Sentry errors count**

Реагировать на проблемы

Алерты

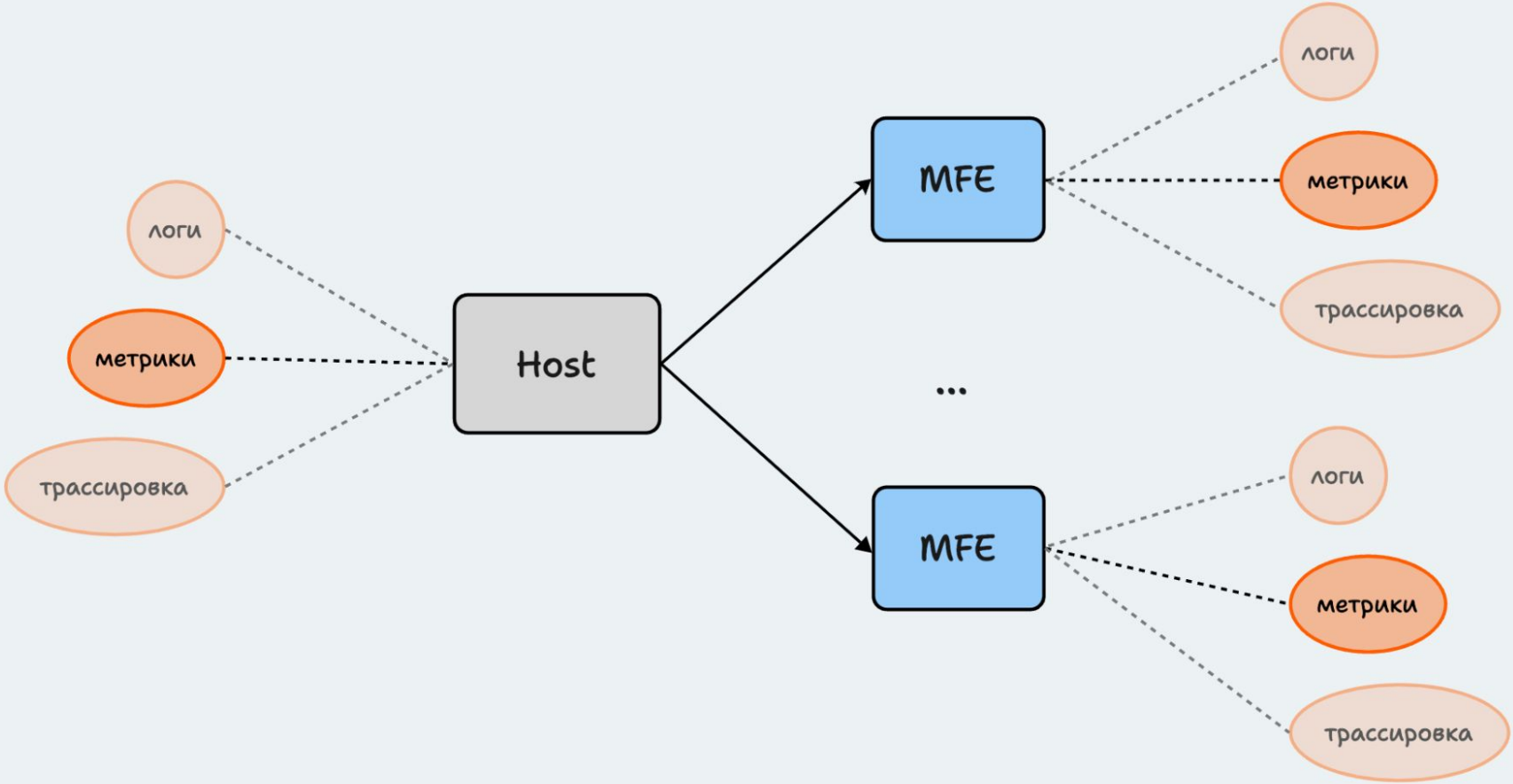
- **Sentry errors count spike**

Итого. Логи

Основные принципы

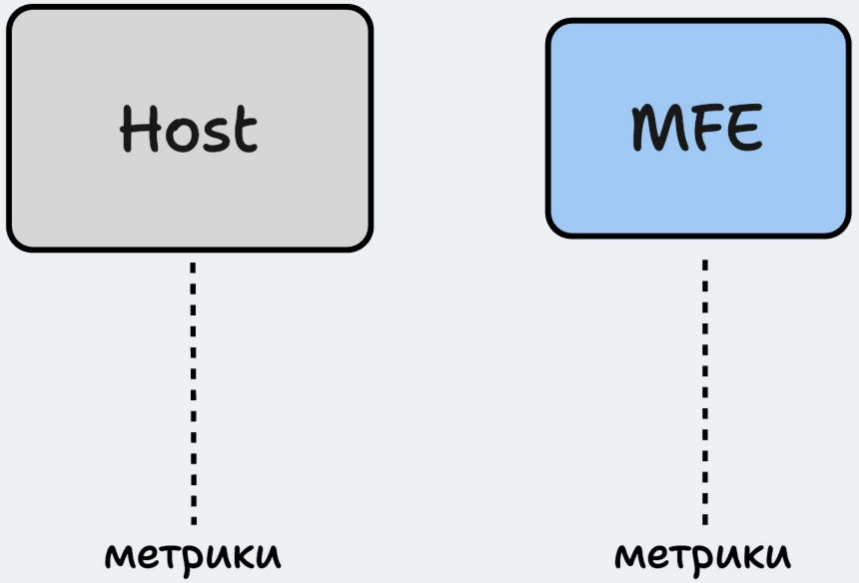
- Пишем логи микрофронтенда из сервиса, в котором он находится.
- Используем инструменты, принятые в компании.
- Используем Sentry Microfrontends для роутинга ошибок.
- Исключаем ошибки, которые невозможно исправить.

Observability. Метрики


логи
метрики
трассировка
Host
MFE
...
MFE
логи
метрики
трассировка
логи
метрики
трассировка

Метрики
Откуда писать

Метрики по микрофронтендам также отправляем из самого сервиса.



Метрики микрофронтенда

Что писать

1 **RPM (Requests Per Minute)**

2 **RT (Response Time)**

3 **HTTP Status Codes**

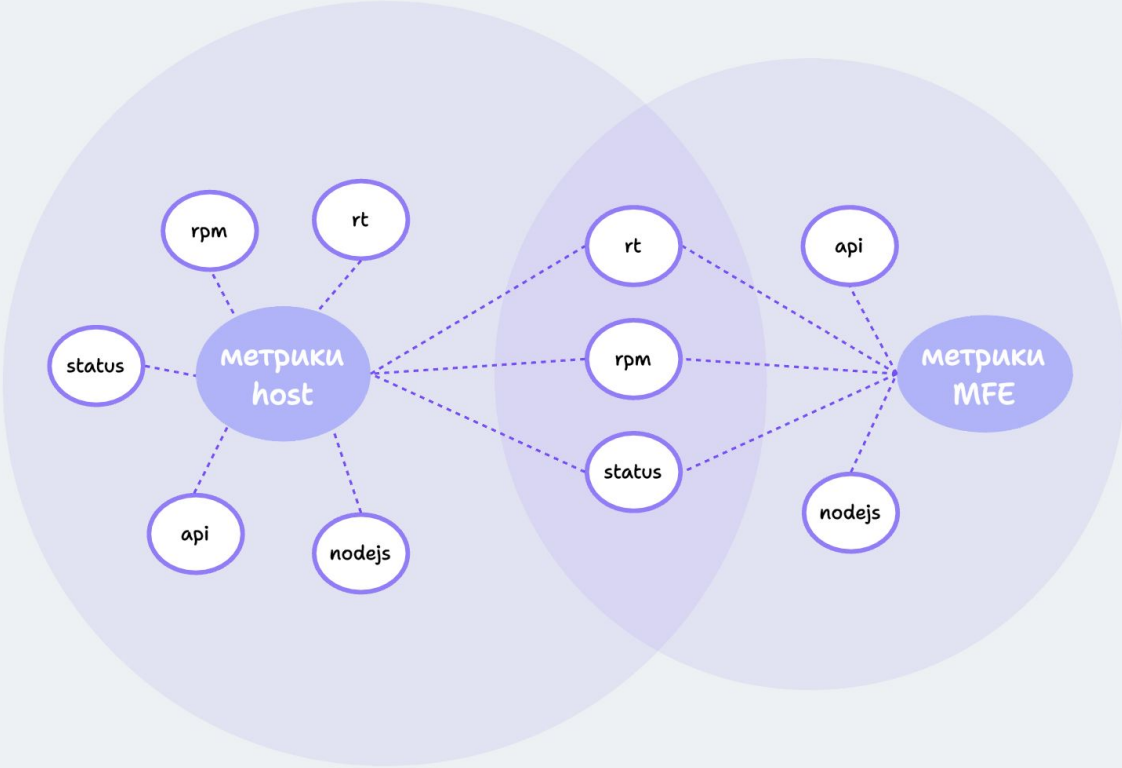
4 **API HTTP Status Codes**

5 **Node Metrics** (CPU%, Mem%, Heap, Event Loop Delay)

Метрики хоста
Что писать

Со стороны хоста пишем метрики микрофронтендов:

- **RPM;**
- **RT;**
- **HTTP Status Codes.**



Метрики

Как выглядит метрика

Пишем из микрофронтенда:

`env.$env.apps.$serviceName.mfe.$mfeName.render.$sr.rt.$status`

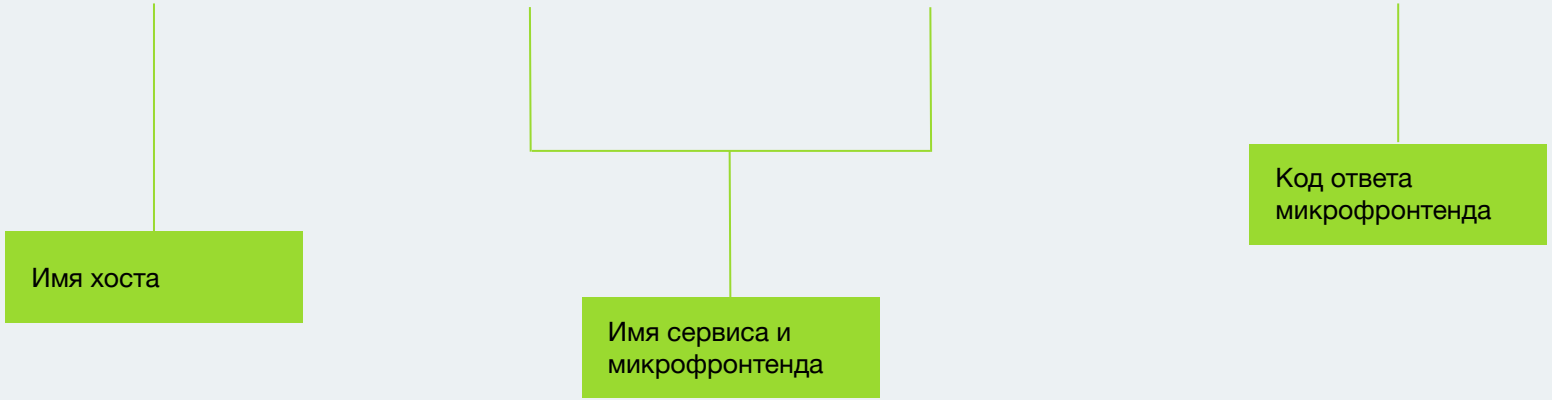
- Обязательно пишем environment
- Имя микрофронтенда может быть не уникально
- SSR / CSR
- Код ответа микрофронтенда

Метрики

Как выглядит метрика

Пишем из хоста:

```
env.$env.apps.$hostName.remote.$serviceName.mfe.$mfeName.render.$sr.rt.$status
```

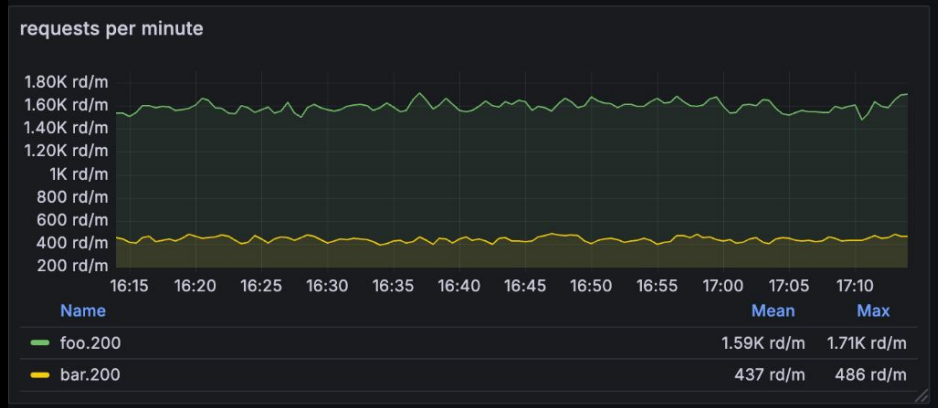


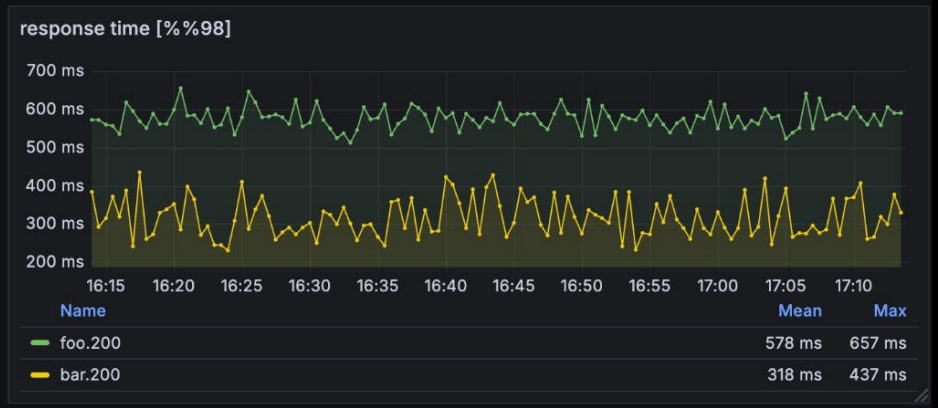
Метрики

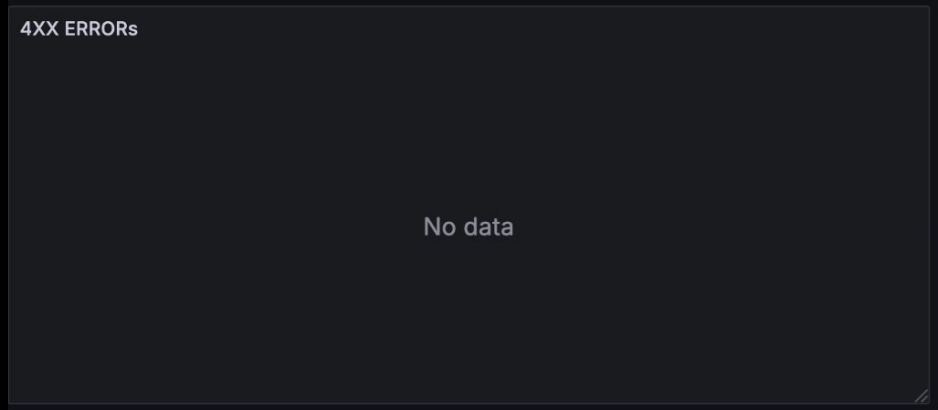
Инструменты

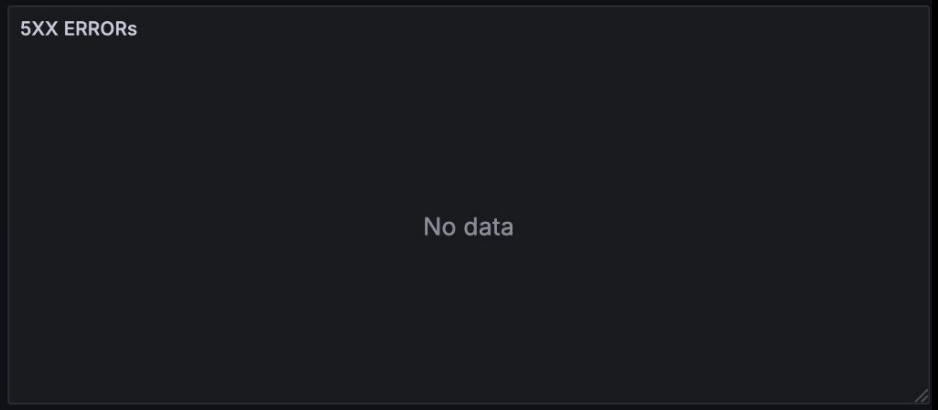
- **Prometheus** — система мониторинга и оповещения для сбора, хранения и анализа метрик в реальном времени.

- **Grafana** — платформа для визуализации, мониторинга и анализа данных в реальном времени.

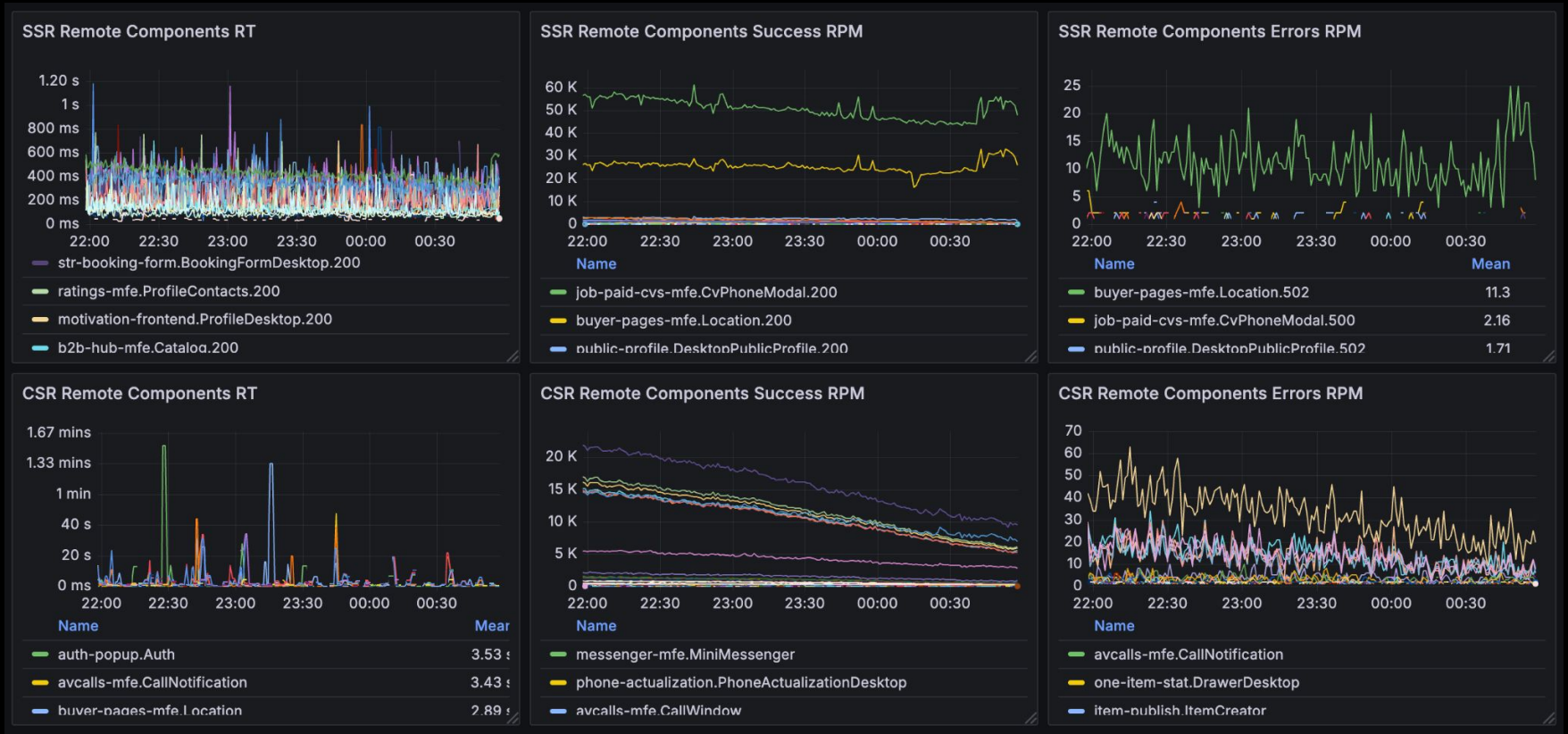




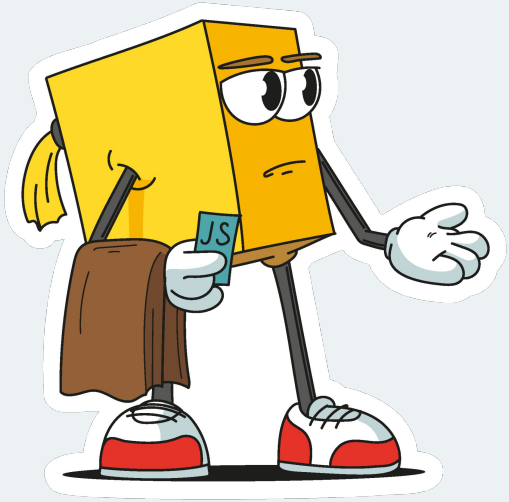




Результат вывода метрик со стороны сервиса с микрофронтендами



Результат вывода метрик со стороны хоста



Например, спросить владельцев, почему так

Метрики
Применение

Предотвращать проблемы

Проберы для канареечных релизов

- 5xx spike
- RT spike
- Mem/CPU climb

Реагировать на проблемы

Алерты

- 5xx spike
- RT spike
- Mem/CPU climb

Измерять успех

NFR

- 5xx count
- RT

Самое сложное – настроить лимиты для пробера, чтобы исключить false positive

Лимиты для проберов

- **5xx spike**
- **New Sentry errors count**

Можем выставить статическое значение, так как по ним редко случаются выбросы.

- **RT spike**
- **Mem/CPU climb**

Будет отличаться в зависимости от времени суток, дня недели и так далее, поэтому статическое значение не подойдет.

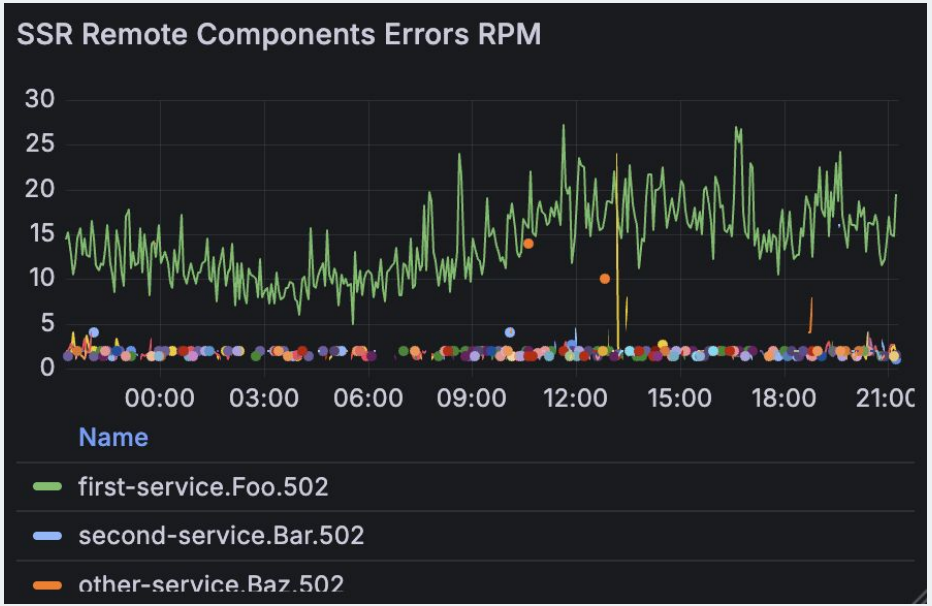
Лимиты для проберов

Пример

- 5xx spike

```
// значение для остановки канарейки
stop: ">50"

// значение для отката канарейки
rollback: ">100"
```

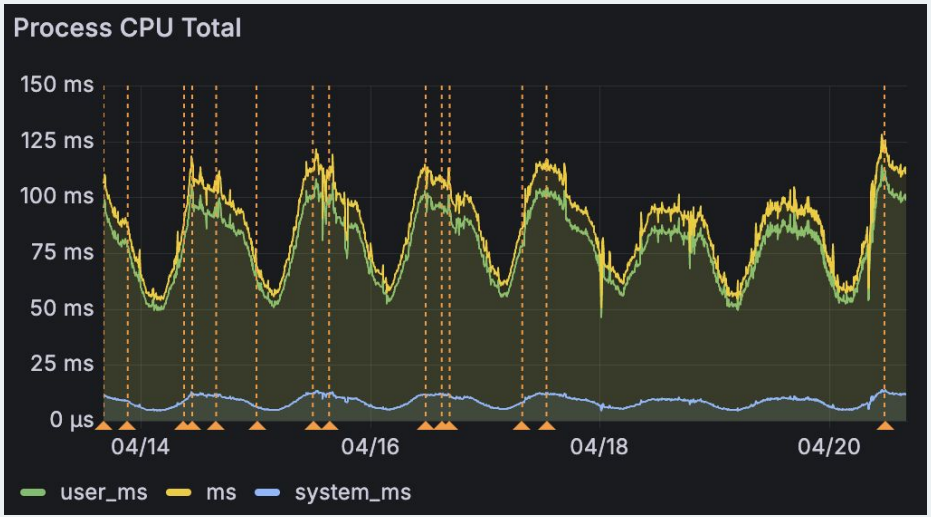


Лимиты для проберов

Пример

- **CPU climb**

Здесь подойдет
baseline
за неделю



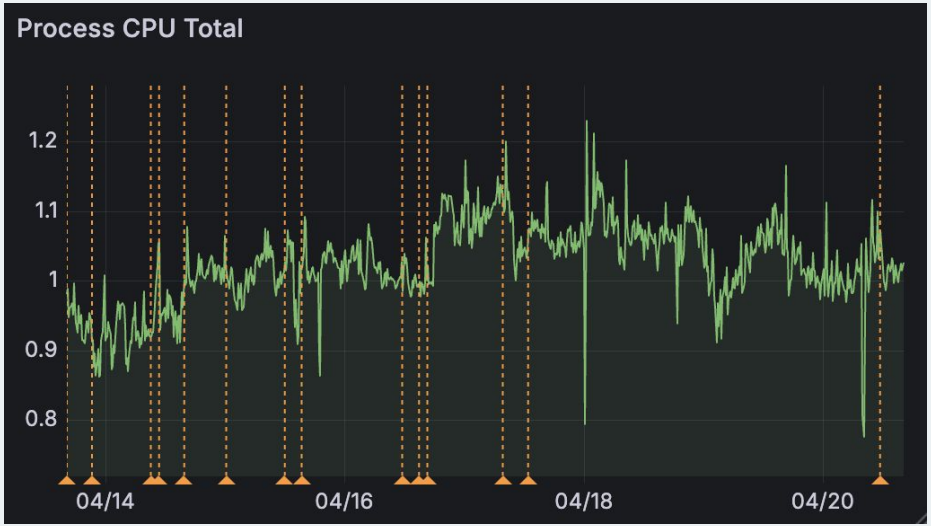
Лимиты для проберов

Пример

- CPU climb

```
// значение для остановки канарейки
stop: ">1.2"

// значение для отката канарейки
rollback: ">1.5"
```

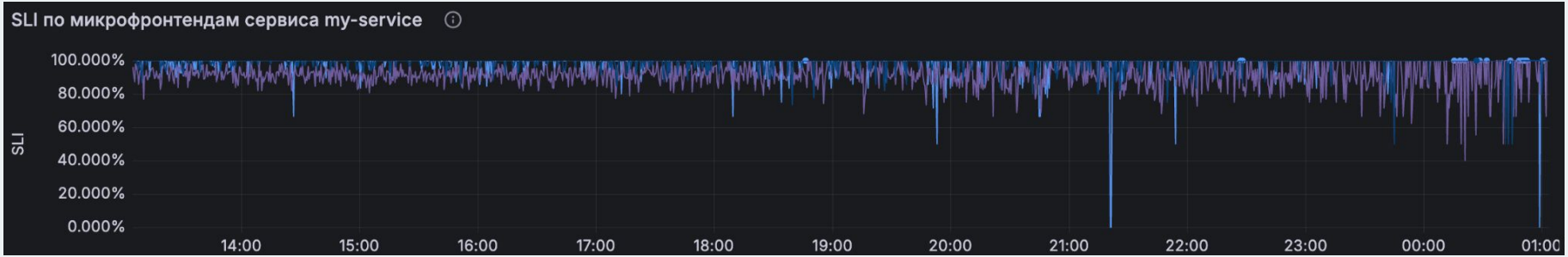


NFR
Пример

- RT
- 5xx count

```
// должен отвечать за 100ms
latency = "100ms"

// ожидаемая нагрузка
rpm = 100
```



Метрики

Пример

Что еще можно писать со стороны хоста и микрофронтендов:

- **перформанс метрики;**
- **продуктовые метрики;**
- **события event-bus.**

Метрики

Высокая кардинальность – характеристика набора данных, указывающая на то, что поле содержит большое количество уникальных значений.

Высокая кардинальность метрики

Пример

Допустим, мы ходим за данными в ручку /user/{userId}.

Высокая кардинальность метрики

Пример

Тогда метрики могут выглядеть так

```
`api./user/123.rt.${status}`
`api./user/456.rt.${status}`
`api./user/789.rt.${status}`
```

Плохо, слишком много значений

Высокая кардинальность метрики

Пример

Правильнее сделать так

```
`api.getUser.rt.${status}`
```

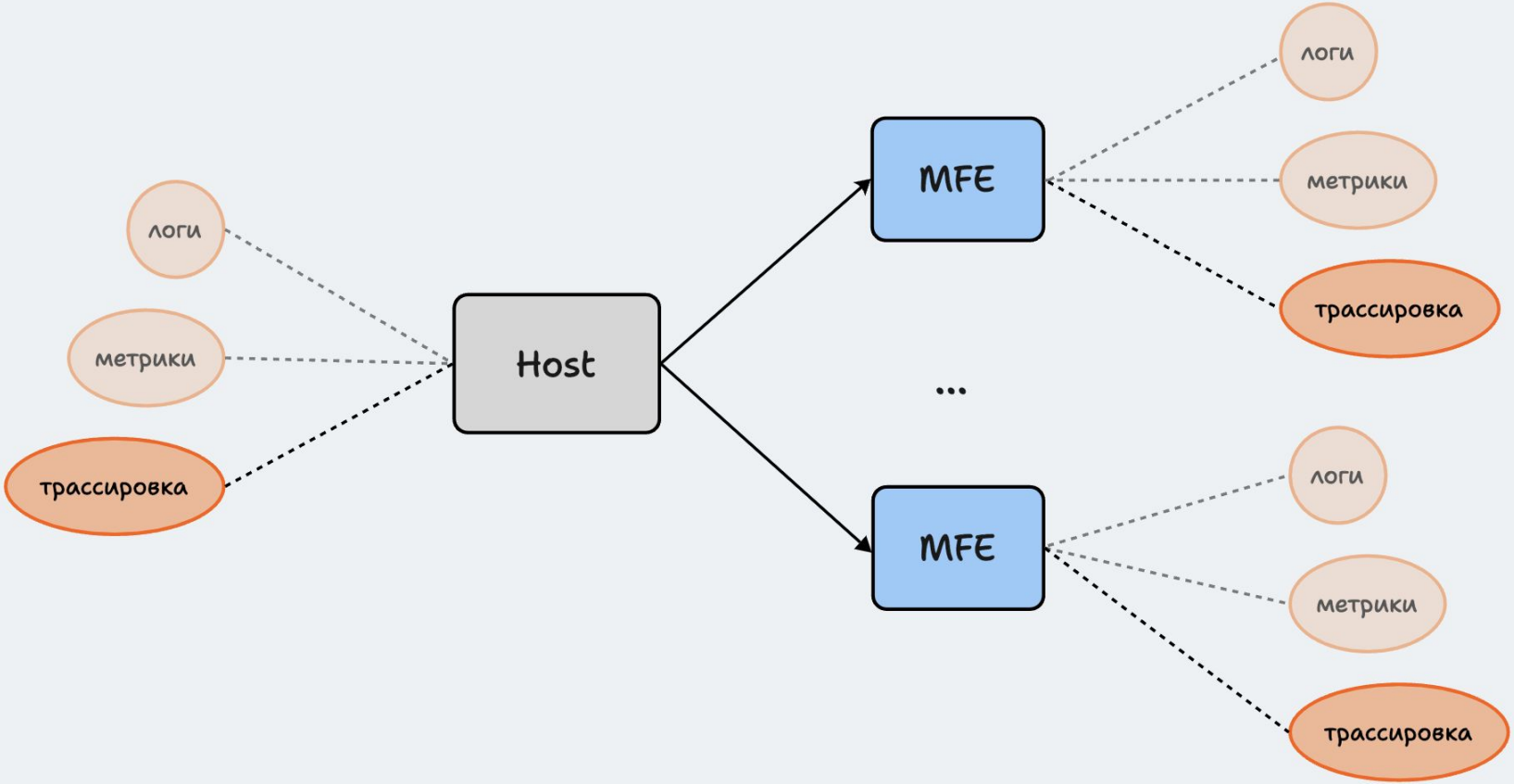
Так как это на самом деле одна ручка

Итого. Метрики

Основные принципы

- Покрываем метриками с обеих сторон.
- Разделяем сервер и клиент.
- Избегаем метрик с высокой кардинальностью.

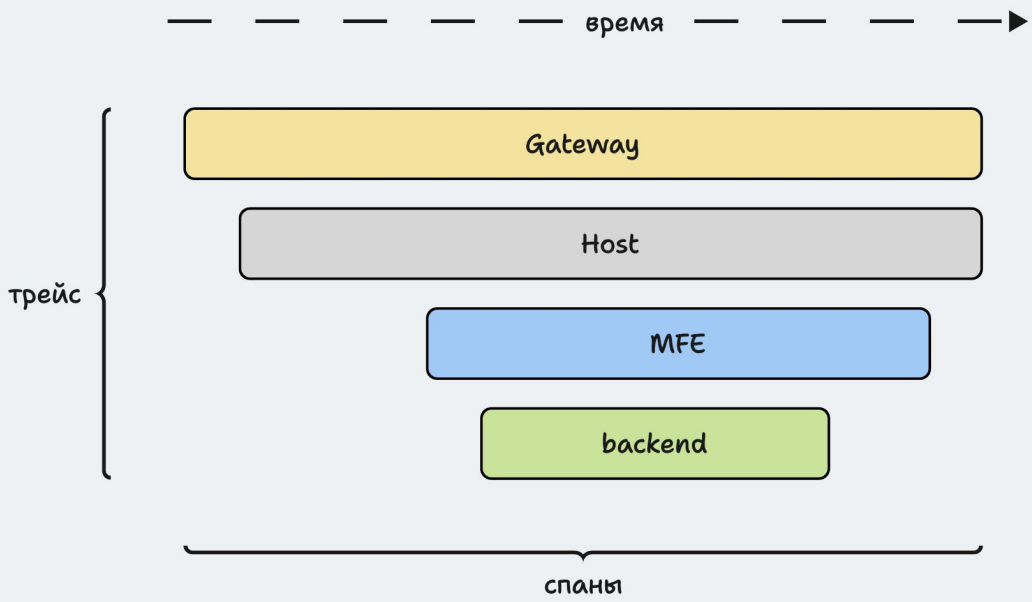
Observability. Трассировка


логи
метрики
трассировка
Host
MFE
...
MFE
логи
метрики
трассировка
логи
метрики
трассировка

Трассировка

Как связываются запросы

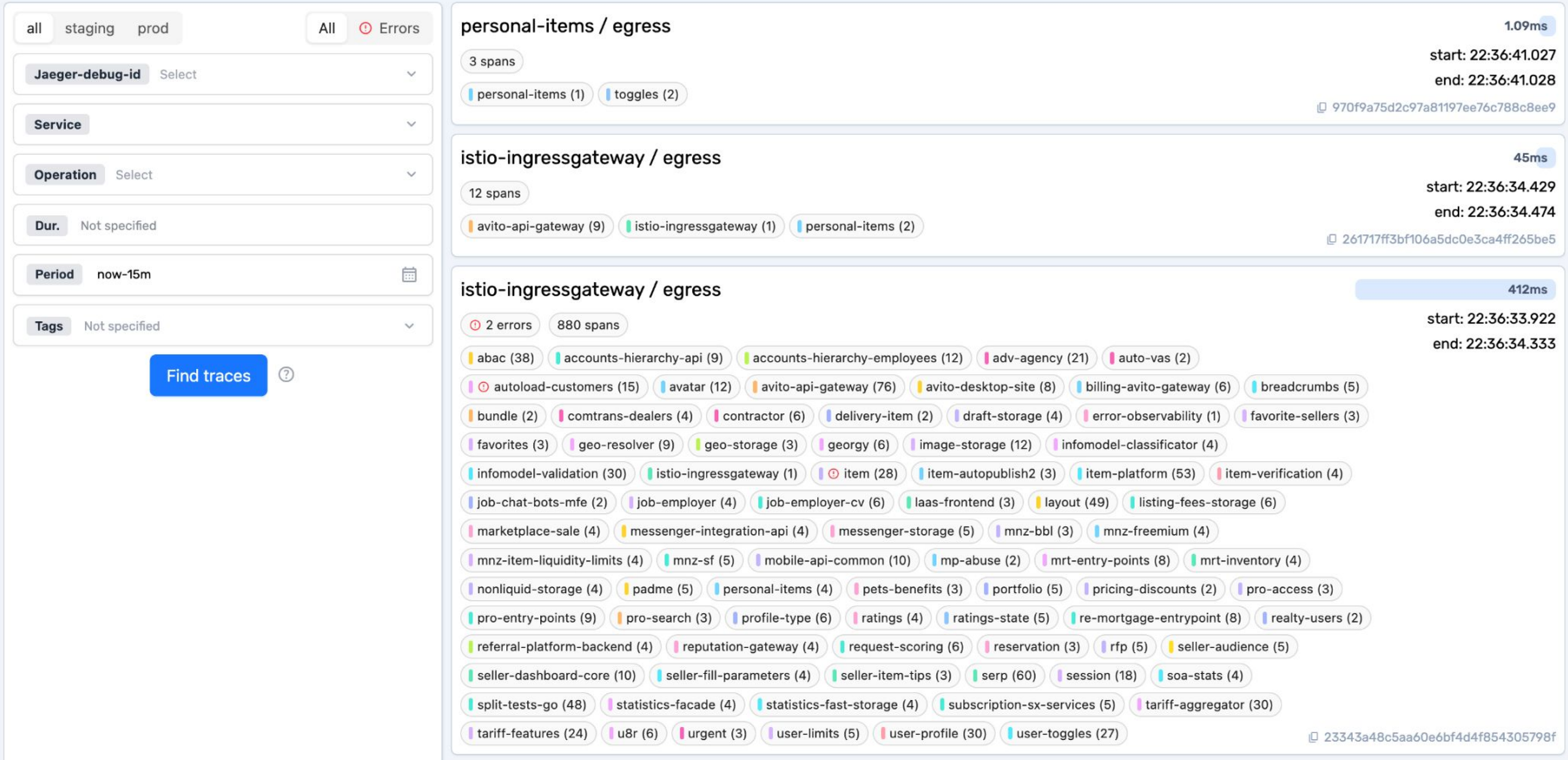
Каждый сервис добавляет свой **«спан»**
с меткой времени, статусом и контекстом.



Трассировка
Инструменты

OpenTelemetry Collector собирает трейсы, агрегирует их и отправляет в централизованную систему хранения.

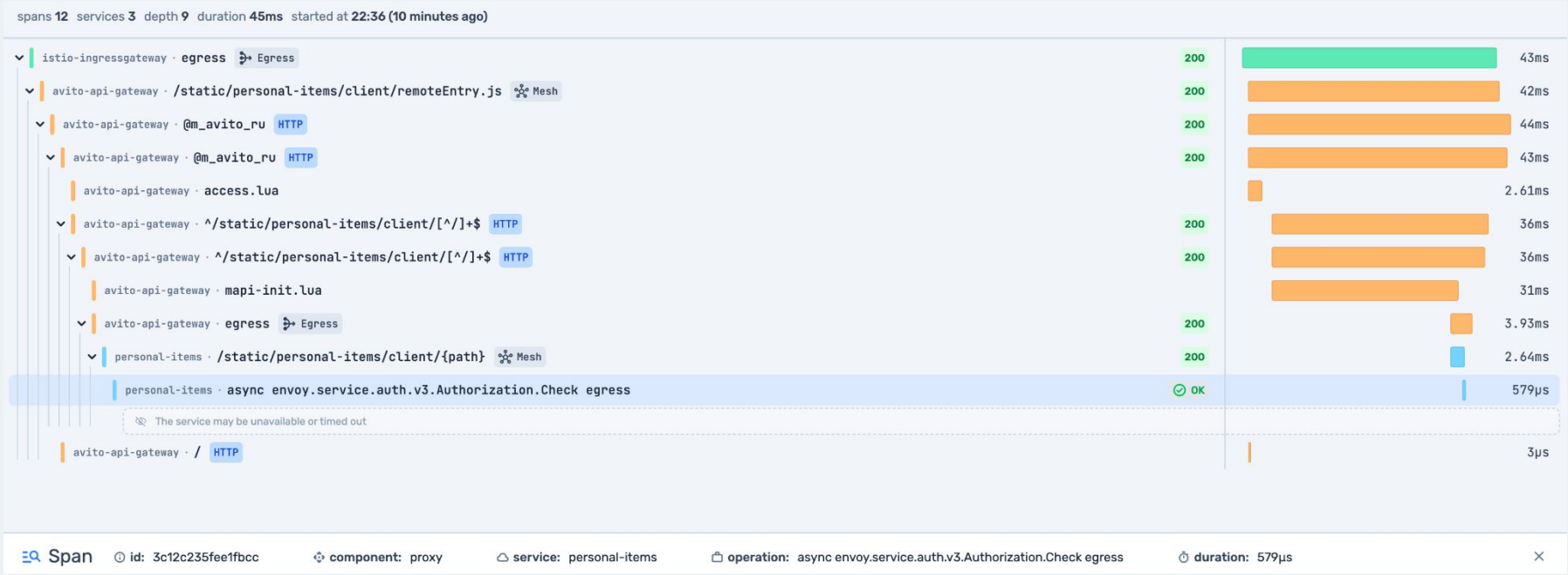
Jaeger используется в качестве бэкенда для хранения и визуализации трассировок.



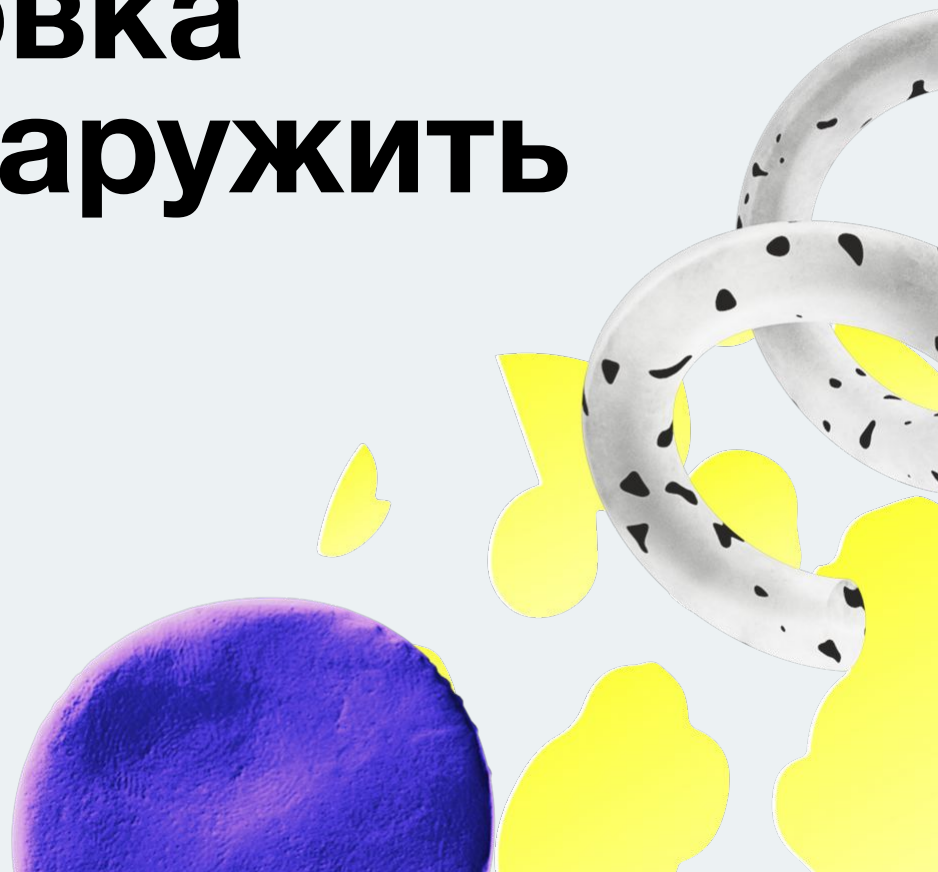
Пример визуализации трассировок

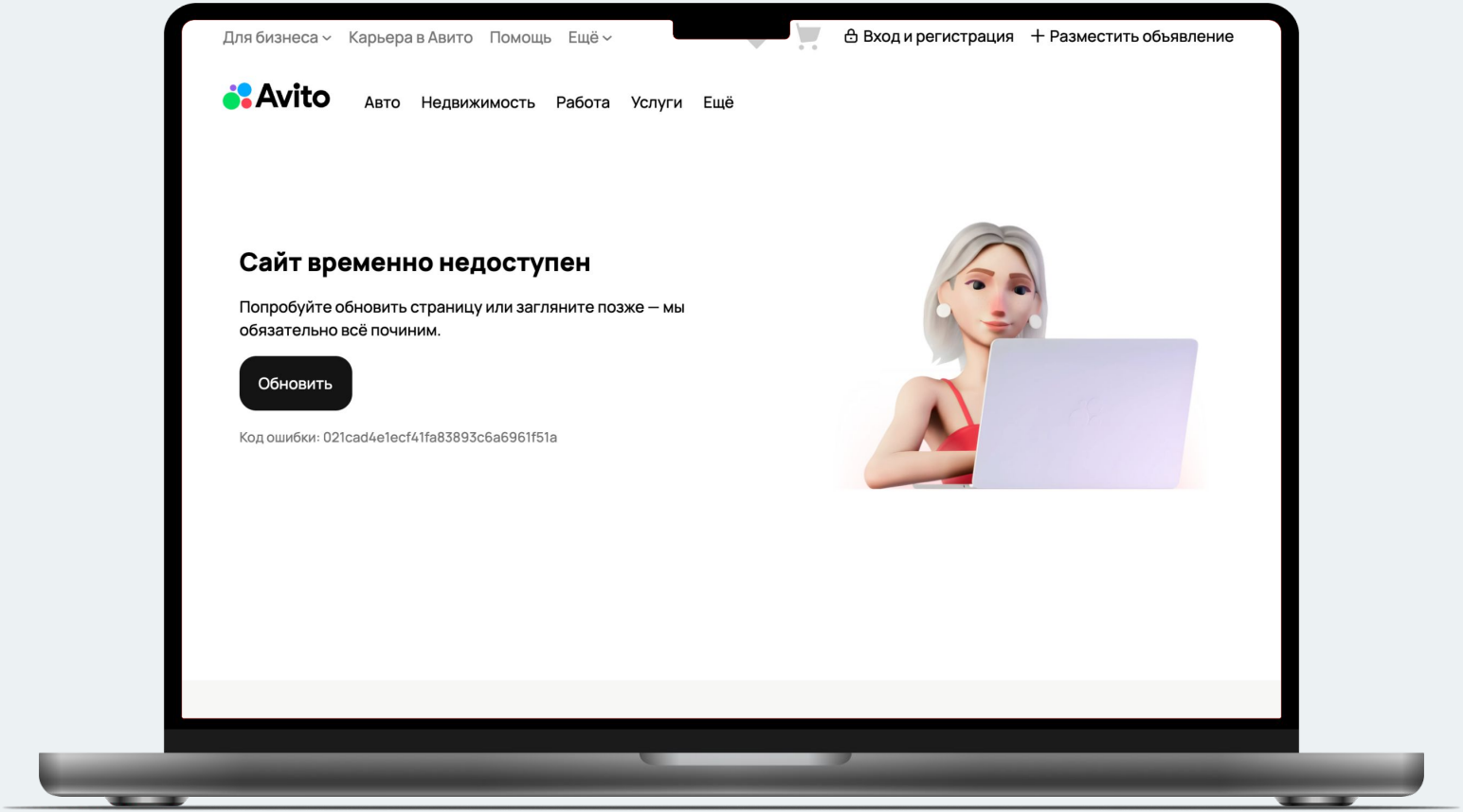
Трассировка

Пример:



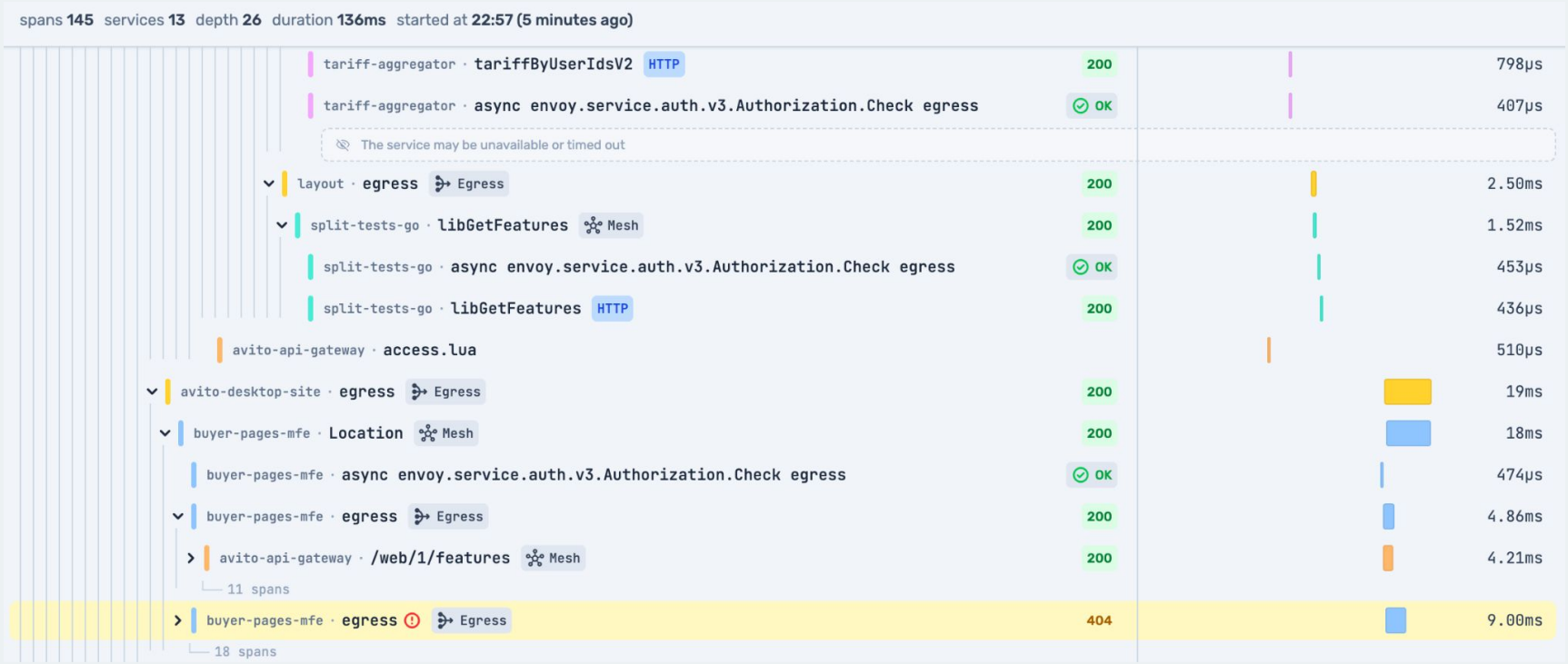
Как трассировка помогает обнаружить проблему




Для бизнеса
Карьера в Авито
Помощь
Ещё
Вход и регистрация
Разместить объявление
Avito
Авто
Недвижимость
Работа
Услуги
Ещё
Сайт временно недоступен
Попробуйте обновить страницу или загляните позже — мы обязательно всё починим.
Обновить
Код ошибки: 021cad4e1ecf41fa83893c6a6961f51a

Трассировка

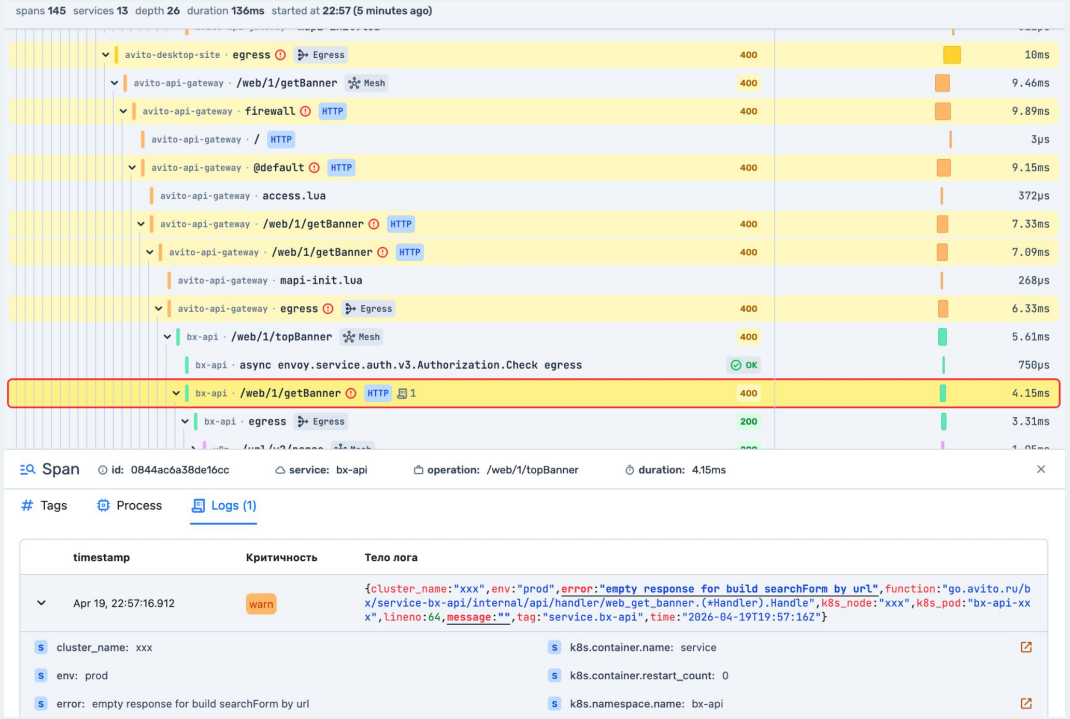
Пример:



Трассировка

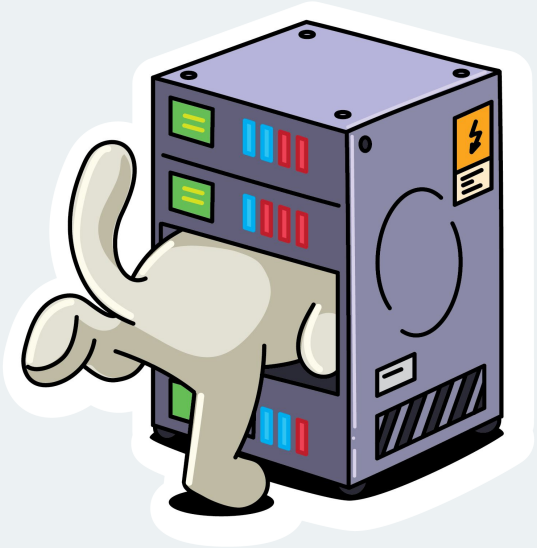
Пример:

Ошибка на бэкенде



Итого. Трассировка

- Помогает в отладке микрофронтендов.
- Позволяет обнаружить долгие ответы.

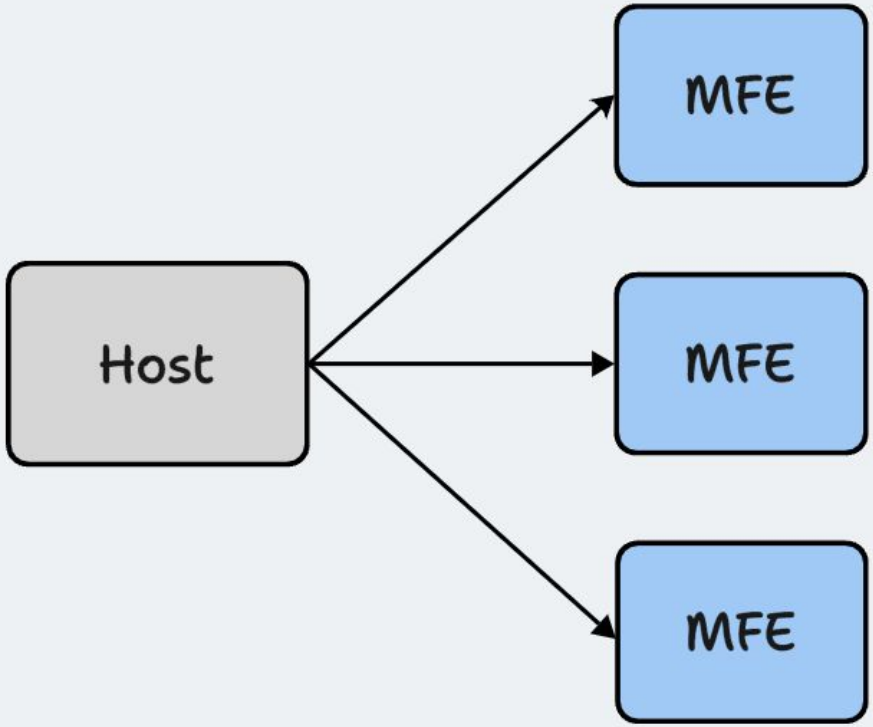


Главное – всё это мы пишем «из коробки»!

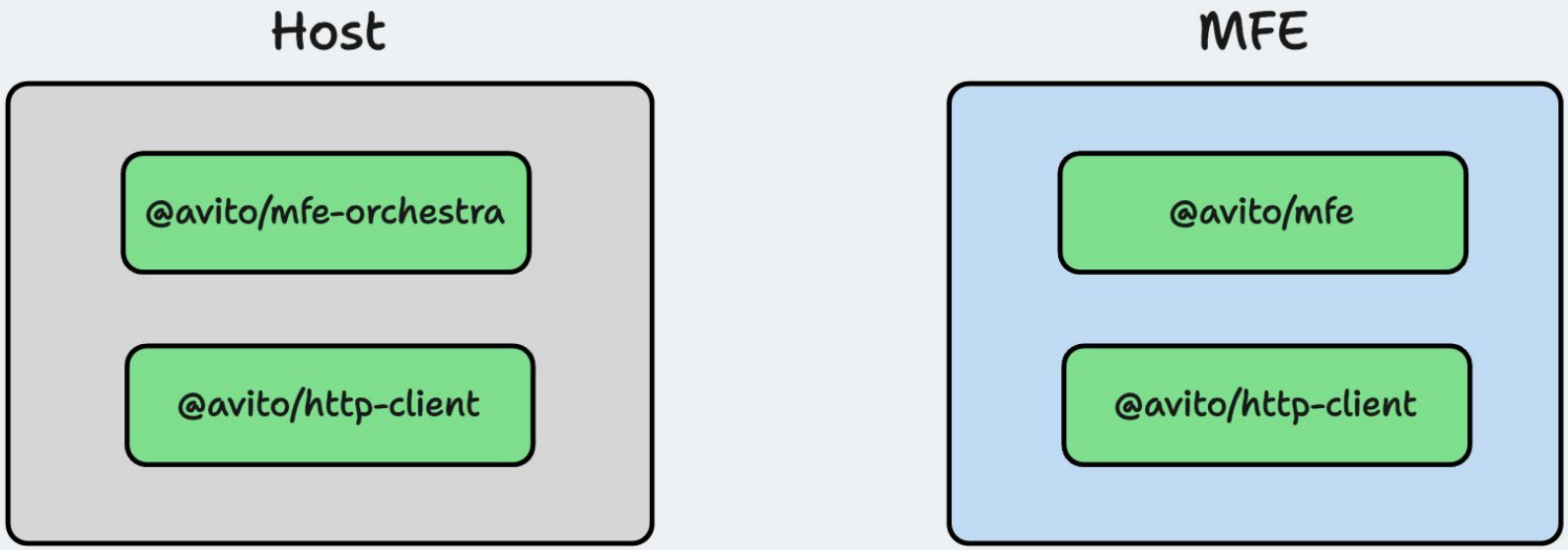
Разберём, как мы добились такого результата

Реализация платформенного решения

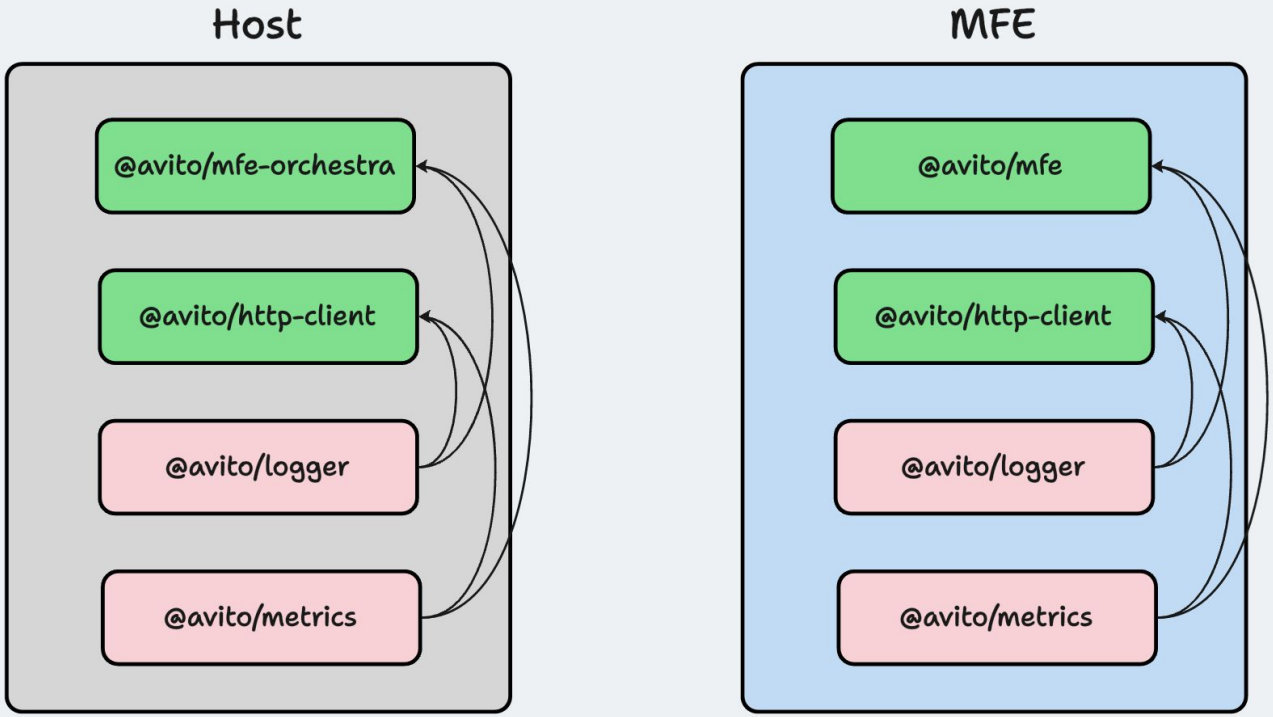
Архитектура микрофронтендов



Как это выглядит на уровне ниже



Решение для логов и метрик



Решение для логов и метрик

`env.$env.apps.$serviceName.mfe.$mfeName.render.$sr.rt.$status`

`env.$env.apps.$hostName.remote.$serviceName.mfe.$mfeName.render.$sr.rt.$status`

`env.$env.apps.$serviceName.mfe.$mfeName.api.$sr.rt.$status`

Решение для логов и метрик

Реализуем платформенные библиотеки **logger** и **metrics**

```
// @avito/logger
Logger.init({
    sentryDsn: process.env.SENTRY_DSN,
    environment: process.env.APP_ENV,
    release: process.env.APP_RELEASE
});

// @avito/metrics
Metrics.init({
    prefix: process.env.APP_NAME
});
```

Переменные подставляются во время деплоя

Создание сервиса

avito service create

Микрофронтенды › SuperFeature

SuperFeature

Grafana Logs Stash Sentry

Основное

О микрофронтенде

Сервис	new-functionality
Описание	Сервис с новой бизнес-функциональностью

Добавление базовых метрик

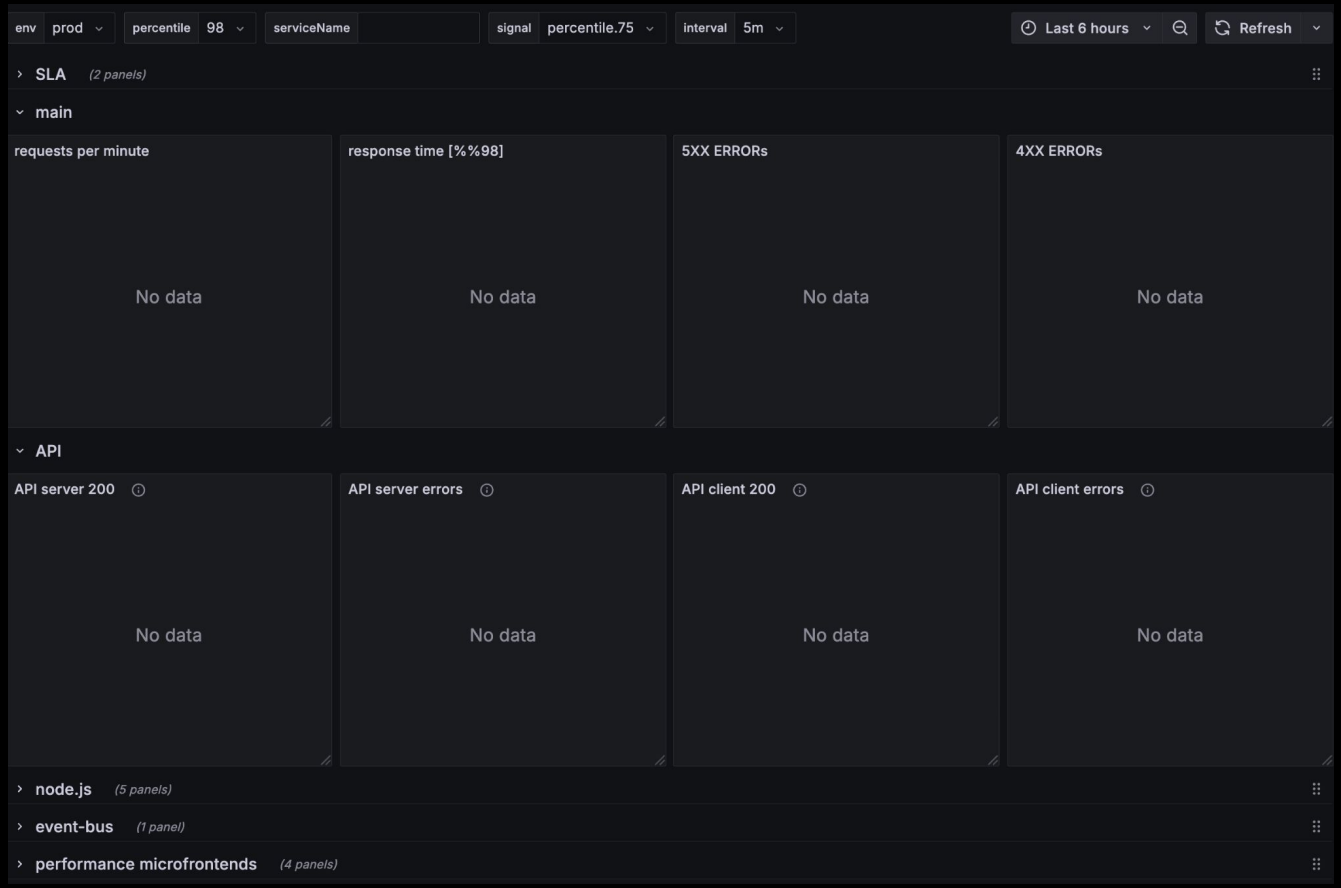
Через Grafana library panels

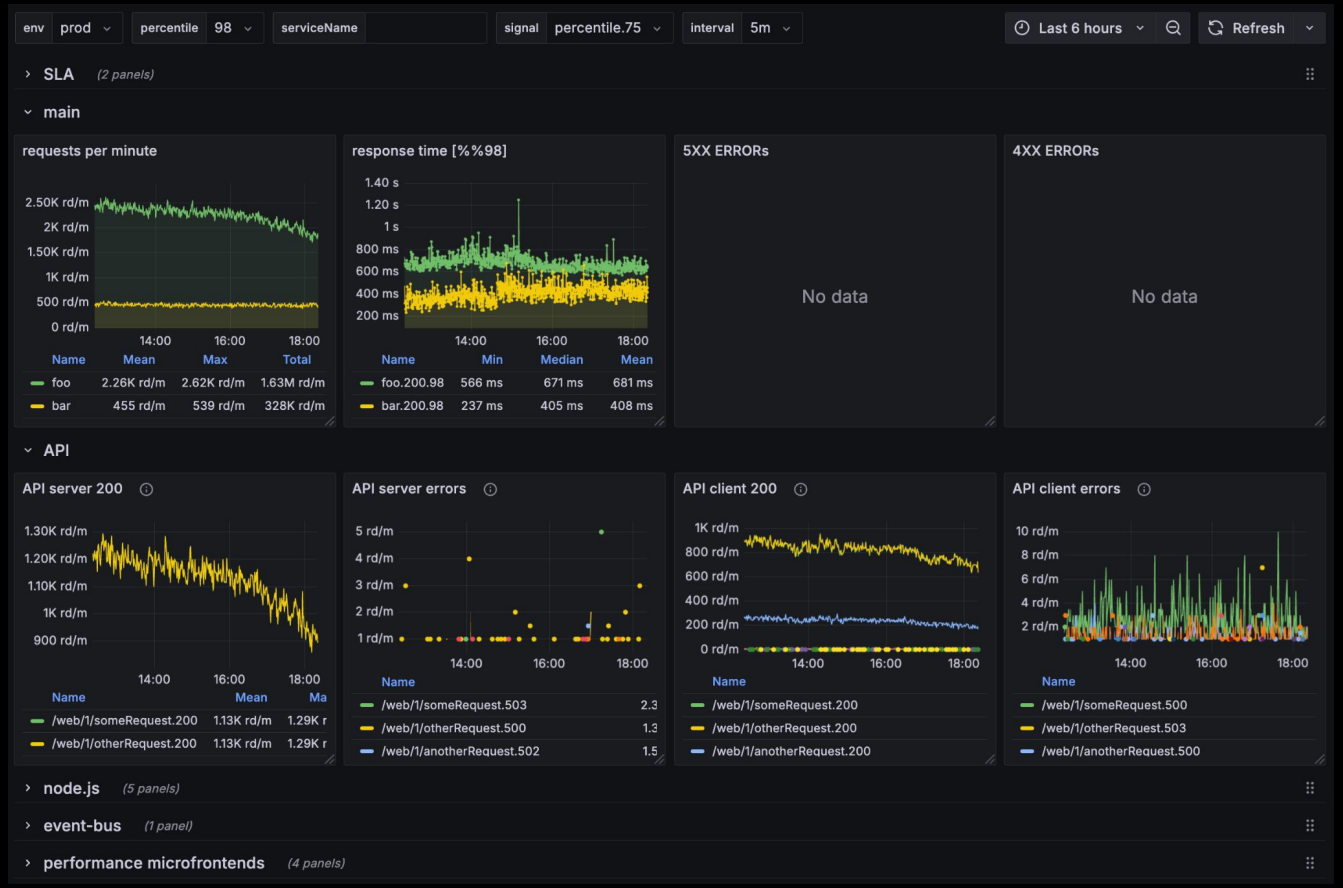
01 Упростить старт

Платформенные метрики не нужно добавлять вручную – они будут доступны сразу.

02 Доставлять обновления

Путей метрик во все микрофронтенды при необходимости.


env prod
percentile 98
serviceName
signal percentile.75
interval 5m
Last 6 hours
Refresh
SLA (2 panels)
main
requests per minute
No data
response time [%%98]
No data
5XX ERRORs
No data
4XX ERRORs
No data
API
API server 200
No data
API server errors
No data
API client 200
No data
API client errors
No data
node.js (5 panels)
event-bus (1 panel)
performance microfrontends (4 panels)


env prod
percentile 98
serviceName
signal percentile.75
interval 5m
Last 6 hours
Refresh
SLA (2 panels)
main
requests per minute
2.50K rd/m
2K rd/m
1.50K rd/m
1K rd/m
500 rd/m
0 rd/m
14:00
16:00
18:00
Name Mean Max Total
foo 2.26K rd/m 2.62K rd/m 1.63M rd/m
bar 455 rd/m 539 rd/m 328K rd/m
response time [%%98]
1.40 s
1.20 s
1 s
800 ms
600 ms
400 ms
200 ms
14:00
16:00
18:00
Name Min Median Mean
foo.200.98 566 ms 671 ms 681 ms
bar.200.98 237 ms 405 ms 408 ms
5XX ERRORs
No data
4XX ERRORs
No data
API
API server 200
1.30K rd/m
1.20K rd/m
1.10K rd/m
1K rd/m
900 rd/m
14:00
16:00
18:00
Name Mean Ma
/web/1/someRequest.200 1.13K rd/m 1.29K r
/web/1/otherRequest.200 1.13K rd/m 1.29K r
API server errors
5 rd/m
4 rd/m
3 rd/m
2 rd/m
1 rd/m
14:00
16:00
18:00
Name
/web/1/someRequest.503 2.3
/web/1/otherRequest.500 1.3
/web/1/anotherRequest.502 1.5
API client 200
1K rd/m
800 rd/m
600 rd/m
400 rd/m
200 rd/m
0 rd/m
14:00
16:00
18:00
Name
/web/1/someRequest.200
/web/1/otherRequest.200
/web/1/anotherRequest.200
API client errors
10 rd/m
8 rd/m
6 rd/m
4 rd/m
2 rd/m
14:00
16:00
18:00
Name
/web/1/someRequest.500
/web/1/otherRequest.503
/web/1/anotherRequest.500
node.js (5 panels)
event-bus (1 panel)
performance microfrontends (4 panels)

Платформенное решение позволило

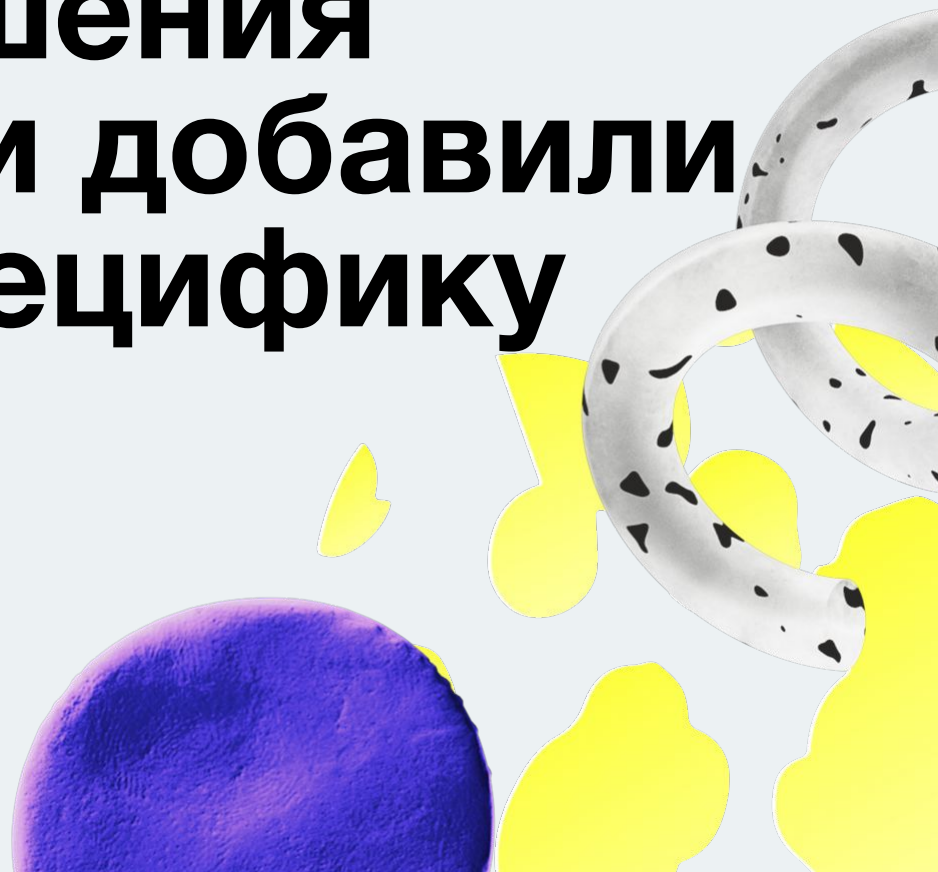
Упростить настройку observability с нуля
благодаря тому, что основное пишем «из коробки».

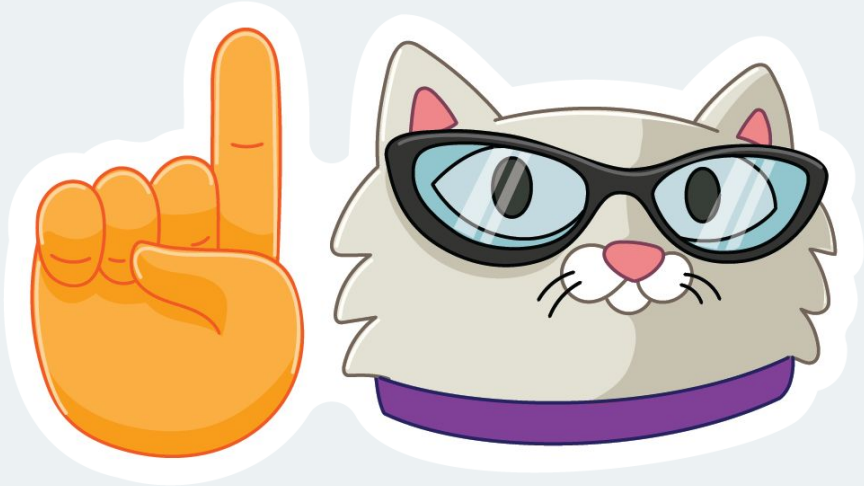
Упростить погружение в чужой проект
благодаря унифицированности.

Нивелировать риск забыть об observability
благодаря тому, что в базовом виде оно всегда есть.

Итоги

Мы взяли решения для бэкенда и добавили фронтенд-специфику





Невозможно предотвратить все ошибки.
Но можно построить систему, которая будет работать тогда, когда ошибаются люди

Контакты

Напишите, если у вас остались вопросы:

Дарья Саенко
Frontend Engineer

dariassaenko@gmail.com